SPECIAL

トランジスタ技術

SPECIAL No.45

特集 PC98シリーズのハードとソフト

386＆486マシンを使いこなす



エレクトロニクスの基礎と実用技術を濃縮したフィールド・ワーク・マガジン

トランジスタ技術SPECIAL

季刊●B5判●定価：1～33定価1,540円 34～45定価1,600円 46～49定価1,690円 50以降定価1,800円

1 個別半導体素子 活用法のすべて 基礎からマスタするダイオード，トランジスタ，FETの実用回路技術	21 ディジタル・オーディオ技術の基礎と応用 最もポピュラーな最新技術を理解しよう	38 Z80システム設計完全マニュアル 周辺I/Oボードの設計とマイコン・システムの開発
3 PC9801と拡張インターフェースのすべて 16ビット・パソコンを使いこなすためのハード＆ソフト	22 ディジタル回路ノイズ対策技術のすべて TTL/CMOS/ECLの活用法と誤動作/トラブルへの処方	39 A-Dコンバータの選び方・使い方のすべて アナログ信号をディジタル処理するための基礎技術
4 C-MOS標準ロジックIC活用マニュアル 実験で学ぶ4000B/4500B/74HCファミリ	23 回路デザイナのためのPLD最新活用法 PLDのプログラミング法からPALライタの製作まで	40 電子回路部品の活用ノウハウ 機器の性能と信頼性を支える受動部品の使い方
5 画像処理回路技術のすべて カメラとビデオ回路，パソコンと隔合させる	24 Cによる組み込み機器用プログラミング 16ビットCPUによるメカトロニクス入門 〈在庫僅少〉	41 実験で学ぶOPアンプのすべて 汎用OPアンプから高性能OPアンプまで
6 Z80ソフト＆ハードのすべて 基礎からマクロ命令を使いこなすまでのノウハウを集大成	25 最新マイコン・メモリ・システム設計法 DRAM，SRAMの動作からデュアル・ポートRAM，FIFOの活用まで	42 高速ディジタル回路の測定とトラブル解析 ハイスピード・ディジタル信号を高周波と捉らえる
8 データ通信技術のすべて シリアル・インターフェースの基礎からモデムの設計法まで	26 68000ソフト＆ハードのすべて 実用ライブラリの作成と便利チップ68301/68303の活用技術	43 Cによるマイコン制御プログラミング 86系ペリフェラルを中心とした
9 パソコン周辺機器インターフェース詳解 セントロニクス/RS-232C/GPIB/SCSIを理解するために	27 ハードディスクとSCSI活用技術のすべて 本格活用のためのハード＆ソフトのすべてを詳解	44 フィルタの設計と使い方 アナログ回路のキーポイントを探る
10 IBM PC&80286のすべて 世界の標準パソコンとマルチタスクの基礎を理解する	28 最新・電源回路設計技術のすべて 3端子レギュレータから共振型スイッチング電源まで	45 PC98シリーズのハードとソフト 386＆486マシンを使いこなす！
11 フロッピ・ディスク・インターフェースのすべて 需要の急増するFDDシステムの基礎から応用まで	29 マイコン独習Z80完全マニュアル 手作りの原点から実用ソフトの作成まで	46 アナログ機能ICとその使い方 民生用AV機器からマルチメディア分野で活躍する
12 入門ハードウェア 手作り測定器のすすめ 電子回路設計の基礎と実践へのアプローチ 〈在庫僅少〉	30 ニュー・メディア時代のデータ通信技術 赤外線，無線通信技術からLAN，光ファイバを用いた高速通信技術まで	47 高周波システム＆回路設計 通信新時代の回路技術とシステム設計
13 シミュレータによる電子回路理論入門 コンピュータを使ったアナログ回路設計の手法を理解するために	31 基礎からのビデオ信号処理技術 複合映像信号の理解からハイビジョン信号の捉え方まで	48 作れば解るCPU ロジックICで実現するZ80とキャッスル・マシン
14 技術者のためのCプログラミング入門 MS-C，Quick C，Turbo Cによるソフトウェア設計のすべて	32 実用電子回路設計マニュアル アナログ回路の設計例を中心に実用回路を詳述	49 徹底解説 Z80マイコンのすべて Z80 CPUの概要から周辺LSIの活用法，ICEのデバックまで
15 アナログ回路技術の基礎と応用 計測回路技術のグレードアップをめざして 〈在庫僅少〉	33 オプト・デバイス応用回路の設計・製作 光素子を使いこなすための製作ドキュメント	50 フレッシャーズのための電子工学講座 電磁気学の基礎から電子回路の設計，製作までをやさしく解説
16 A-D/D-A変換回路技術のすべて アナログとディジタルを結ぶ最新回路設計ノウハウ	34 つくるICエレクトロニクス 機能ICを使って実用機器を作ろう 〈在庫僅少〉	51 データ通信技術基礎講座 RS232Cの徹底理解からローカル通信の実用技術まで
17 OPアンプによる回路設計入門 アナログ回路の誤動作とトラブルの原因を解く	35 C言語による回路シミュレータの製作 Quick Cでのプログラミングとフィルタ回路の解析	52 ビデオ信号処理の徹底研究 映像信号の基礎から高画質化のためのディジタル信号処理の方法まで
19 PC9801計測インターフェースのすべて オリジナル拡張ボードでパソコンを実践活用しよう	36 基礎からの電子回路設計ノート トランジスタ回路の設計からビデオ画像の編集まで	53 パソコンによる計測・制御入門 研究室や実験室で必要なデータ収集のノウハウを基礎から解説
20 アナログ回路シミュレータ活用術 ゲーム感覚の回路設計を体験しよう	37 実用電子回路設計マニュアルII 豊富な回路設計例から最適設計を学ぼう	

CQ出版社 〒170 東京都豊島区巣鴨1-14-2 販売部☎03-5395-2141 振替 00100-7-10665

トランジスタ技術 SPECIAL No.45

CONTENTS

特集 PC98シリーズのハードとソフト

386&486マシンを使いこなす

吉田 功

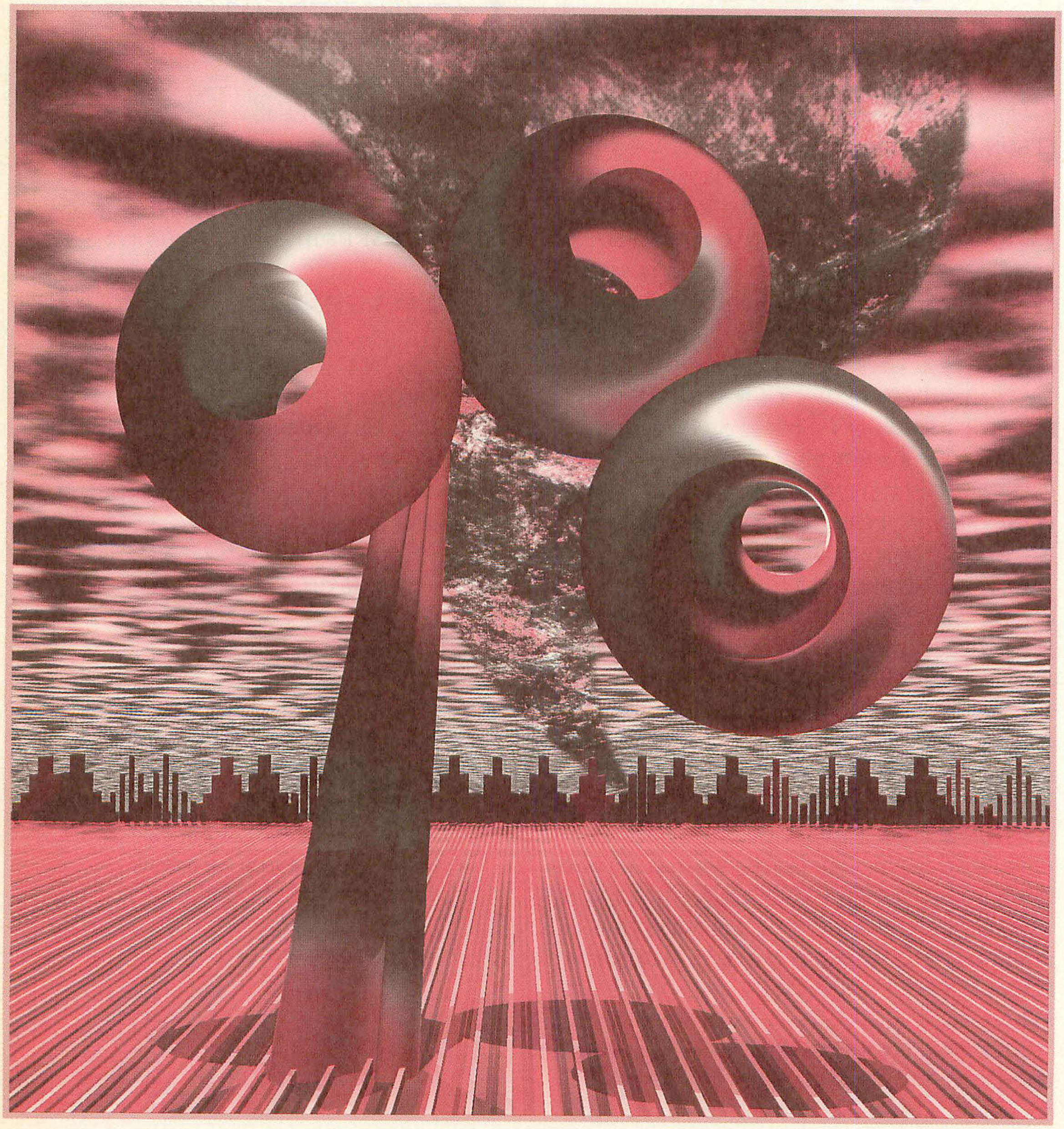
表紙デザイン/㈱ザ・ステューディオ・トウキョウ・ジャパン　本文イラスト/神崎真理子

特集　PC98シリーズのハードとソフト

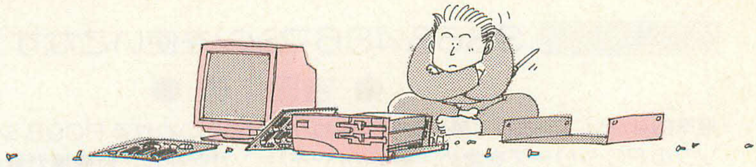
386&486マシンを使いこなす

● 吉田　功 ●

標準的なパソコンとして最も数多く使われているといわれるPC98シリーズには，エプソンのPCシリーズを含め，多くのバージョンが存在し，新機種が発売され続けています．基本的なシステム構成は変わらないものの，使う側にとってバージョンの違いを知ることは重要です．ここでは最新の98シリーズのハードとソフトを詳解しています．



1 PC98シリーズのシステム構成



98シリーズの違い

1982年の秋に，NECから初めてのPC98シリーズであるPC9801が発表されて以来，現在までに非常に多くの機種が発売されてきました．1987年の春には，EPSONからも98互換機としてPC98シリーズと，ほぼ同機能なコンピュータEPSON-PCシリーズが発表され，NEC同様に多くの機種が発売されています．

現在，PC98シリーズは目的別に数種類に分かれます．まず，シリーズの核になっているのはPC9801系列です．ノーマル・モードと呼ばれます．普通，98といえば，この系列のことをさしますし，他の系列の機種でもノーマル・モードを持っています．

次に，そのPC9801シリーズを携帯用に小型化したPC9801ノート・ラップトップ系列で，PC9801のソフトウェアをそのまま動作できるように作られていますので，かなりの部分で互換性があります．例外はPC98LT/HAで，PC9801系列とは画面表示関係が大きく異なり，互換性が少ないものになっています．

また，狭い画面表示を補うために1120×750ドットの表示が可能になったハイレゾ対応機種が1985年に生まれました．名前がPC9801ではなくPC98と呼ばれて差別化されています．初めてのハイレゾ対応機である PC98XA以外の機種では，ハイレゾとノーマルの二つのモードを持ちますので，ノーマル・モードの多くのソフトウェアを，そのまま使用することができます．また，1990年からはPC-H98シリーズと独立したシリーズになりました．

1992年からは，PC9821と呼ばれる新しいシリーズができました．640×480ドット256色の画面を持つシリーズで，マルチメディアや，MS-WINDOWSの使用に有利なように作られています．基本的にはPC9801シリーズと同等で，新しい機能が追加される形になっています．

これらの新しい機能は具体的にハードウェアを直接操作せずに，BIOS等を通して操作するのを前提としているようで，同じ9821の名前が付いていても，ハード的には異なる場合があるようです．

◆ ノーマル・モードとハイレゾ・モード

ノーマル・モードとハイレゾ・モードの違いは，表示画面サイズが変わっただけではなく，メモリ・マップも大きく変更されています．

まず，メイン・メモリ空間(コンベンショナル・メモリ)が128Kバイト増えて768Kバイトになりました．VRAM空間もC0000H～EFFFFH(テキストはE0000H～E3FFFH)になり，一箇所にまとまっています．

I/O関係も若干異なり，マウス・インターフェース等ではアドレスも異なります．プリンタ・インターフェースでは，フル・セントロニクス仕様に拡張されているためにI/Oアドレスは同じものの，内容は大きく違います．他にも細かい部分で，変更されています．

ハイレゾ・モードは，ノーマル・モードより，いろいろなところで優れていますが，NECの販売戦略のためか，価格が全般的に高く設定され，一般的ではありません．筆者もまた，ハイレゾ・マシンには縁遠いためもあって，本誌では，ハイレゾ・モードについては詳しく述べることはできませんでした．

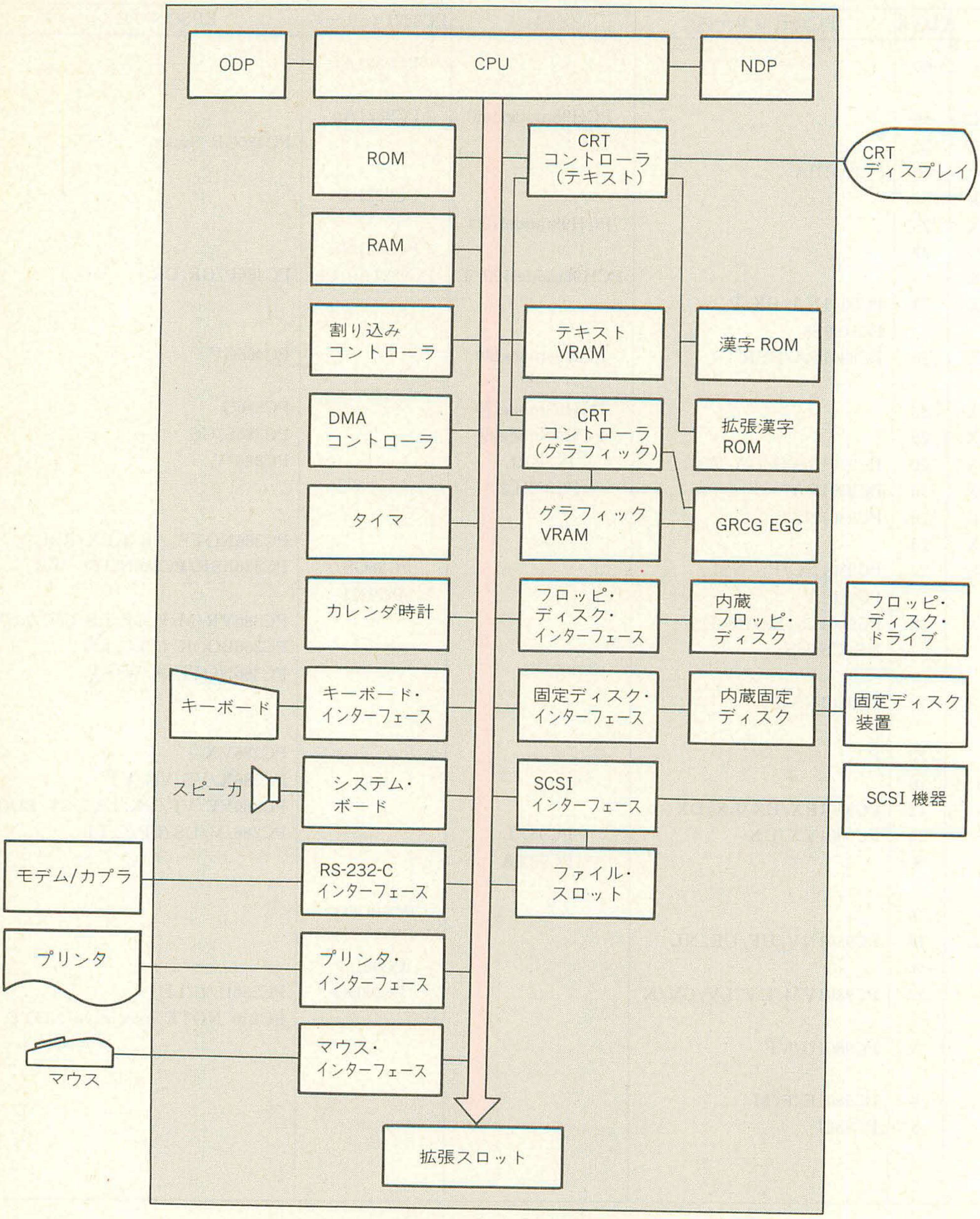
◆ 周辺LSIの違い

PC98シリーズのCPUは，年々新しく違ったものが搭載されてきましたが，CPUの高速化に合わせて，周辺LSIも，図1-1に示すようにグレードアップされてきました．搭載されているLSIは，細かい追加

〈図1-1〉周辺LSIの移り変わり

LSI	主に386までの機種		主に486以降の機種
	NMOS		CMOS
DMAC	μPD8237A	→	μPD71037
SIO	μPD8251A	→	μPD71051
PIT	μPD8253	→	μPD71054
PPI	μPD8255A	→	μPD71055
PIC	μPD8259A	→	μPD71059
FDC	μPD765A	→	μPD72065
GDC	μPD7220	→	μPD72020

〈図 1-2〉 PC98 シリーズのハードウェアのブロック・ダイヤグラム



機能があったり，処理速度の向上がなされていますが，完全上位互換になっています．

LSI の処理速度の向上は，I/O のリカバリ・タイムの短縮に関係してきます．μPD710××を載せている機種では，CPU 速度が上がったにも関わらずリカバリ・タイムが減っています．

また，最近の機種では，周辺 LSI のほとんどが集合化されカスタム・チップになってきました．μPD71055 等の汎用 LSI は，メイン基板からなくなり，CPU とメモリ以外はいくつかのカスタム LSI が載っているだけになっています．

◈ ブロック・ダイヤグラム

PC98 シリーズのハードウェアをブロック図で表してみました．機種によって搭載されている機能が，かなり変わりますが，基本的には図 1-2 のようになります．

CPU

PC98 シリーズには，非常に多くの機種があり，それに搭載されている CPU も数多くあります．最初に

〈図 1-3〉PC98 シリーズに搭載されている CPU

CPU	CLOCK	PC9801 シリーズ	ハイレゾ	PC9821 シリーズ	EPSON-PC シリーズ
Pentium	60			PC9821Af	
80486DX2	66		PCH98model105	PC9821Ap	
80486DX2	50				PC486GR Super
80486DX2	40	PC9801BA			
80486DX	33			PC9821As	
80486DX	25		PCH98model100		
80486SX	33			PC9821Ne	
80486SX	25		PCH98model190/T	PC9821Ae/Ce	PC486P/GR/GR＋
80486SX	20	PC9801NA/BX/P/ PCH98S8			
80486SX	16	PC9801FA/NSR	PCH98model80		PC486GF
80386DX	33		PCH98model70		PC386G
80386DX	25		PCH98model60		PC386S/GS
80386DX	20	PC9801RA21/DA/NX	PC98RL		PC386/V
80386DX	16	PC9801RA	PC98XL2		
80386SL	20	PC9801NST			
80386SX	25				PC386NOTE AR/ARX/ARC
80386SX	20	PC9801NC/FS/NSL/ PC9801T		PC98GS/ PC9821	PC386LSR/PC386NOTE WR
80386SX	16	PC9801LS/ES/RS/DS/ NS/CS/US			PC386VR/M/P/GE/LS/LSC/LSX PC386BOOK L/LC/LX PC386NOTE A/W/AE
80386SX	12	PC9801NS/FX			
80286	20				PC286VX
80286	16				PC286X/VS/VG/VJ
80286	12	PC9801RX/EX/LX/DX			PC286VE/VF/UX/LS/LST/BOOK
80286	10	PC9801VX/UX	PC98XL		PC286/V/US/LF/C/LP
80286	8		PC98XA		
V33A	16			PC98DO＋	
V30HL	16	PC9801NV/UF/UR/NL			
V50	8			PC98LT	
V30	10	PC9801VM/UV/LV/CV/N		PC98DO	PC286U/L/LE PC286 NOTE executive/NOTE F
V30	8	PC9801U/VF			
8086-2	8	PC9801E/F/M			
8086	5	PC9801			

登場した PC9801 に採用された 8086 から，最新の Pentium まで，その実行速度の差は数十倍にも達します．搭載 CPU 別に機種名をまとめてみました(図 1-3)．

◈ CPU 格差による速度差の吸収

同じ PC98 シリーズといえども，CPU の処理速度差が大きく違います．機種によっては，ROM や RAM の速度が CPU に付いてきませんし，拡張スロットの規格は 80286 の 10 MHz の頃から変わっていませんから，CPU がメモリや周辺に合わせてウェイトをかけます．

ウェイトの数は，大まかに分けると図 1-4 に示すように，CPU の種類とクロックの周波数で分けられます．機種によって若干違いがあるものがあります．

◈ 内蔵 RAM と増設 RAM

本体のマザー・ボード自体に初めから付いている内蔵 RAM と，後からメモリ専用スロットに増設する専用拡張 RAM，本体後部の拡張スロットに増設する汎

〈図 1-4〉 CPU の種類とクロック周波数で分けるウェイト

8086	5MHz	8MHz
RAM/ROM	0	1
I/O	1	2

9801/E/F/M

V30	8	10	10	16
内蔵 RAM	1	1	0	0
拡張スロット ROM	2	3(2) UV21	2	6(1) NV, NL
I/O	2	3	3	8(2)

9801U/VF/VM/UV LV/CV/UV/N UF/UR/NV/NL
0.24 2 21 21 11
21 21 22

286	8	10	12
内蔵 RAM	0	0	0
拡張 RAM	1	1	0(専用)
拡張スロット ROM	4	5	6
I/O	3	4	6

386	12	16	20
内蔵 RAM	0	0	0
専用拡張 RAM		0	0
拡張スロット ROM	5	8−10	10−12
I/O	5−6	8	10

486	16MHz 9801FA	20	25	33
内蔵 RAM	0	2/1	2/1	2/1
専用拡張 RAM	1	2	2	2
拡張スロット ROM	9	11	14?	19?
I/O	9	11	14?	19?

用拡張 RAM とがあります．これらは自動的に挿入されるウェイトが変わります．特に CPU に 80386 を搭載している場合は，プロテクト・メモリを有効利用できますので，ウェイト数が少ない専用拡張 RAM が有利です．

◈ CPU に対する命令

80286 以上の CPU を搭載している機種には，CPU に対してのみ行う CPU リセットと，プロテクト・モード ON の操作を行うポートがあります．

CPU リセットは，CPU のみにリセットをかけて，周辺の LSI などへはそのままです．80286 には，8086 互換モードのリアル・モードと，1M バイト以上のメモリ空間を使用することができるプロテクト・モードがありますが，プロテクト・モードからリアル・モードへの移行はソフトウェアではできません．そのため CPU にリセットをかけることで，これを実現しています．80386 以降の CPU では，ソフトウェアでこれらの切り替えができます．

CPU にリセットをかける場合，電源投入時なのか，リセット・スイッチが押されたのか，モード切り替えのために CPU のみにリセットをかけたのかは，システム・ポートのポート C・D_5/D_7を参照すれば知ることができます．

プロテクト・モード ON は，8086 や V30 等のメモリ空間が 1M バイトしかない機種と，80286 以上のメモリ空間が 1M バイト以上ある機種との違いを吸収するもので，プロテクト・モード ON にすることで，1M バイト以上のメモリをアクセスすることができるようになります．

80286 等の 1M バイトを超える CPU では，8086 互換モードでもアドレス・ラインは 1M バイト以上のメモリをアクセスできます．例えば，セグメント＝FFFFH/オフセット＝FFFFH とすれば，ソフトウェア的には，FFFF0H＋0FFFFH＝10FFEFH と，1M バイト＋64K バイトのアクセスが可能となります．

しかし，8086/V30 では，アドレス・ラインが 20 本しかないため，10FFEFH とはならず，0FFEFH となってしまいます．

プロテクト・モード ON は，これらの違いを吸収するために付けられています(図 1-5)．

PC9801RA21/DA/FA 等の機種では，プロテクト・モード ON に加えてプロテクト・モード OFF(図 1-6)も可能です．

◈ プロセッサ ID

80386 以降の CPU では，CPU がリセットされた直後に DX レジスタにプロセッサ ID が入っています．機種によっては，この値をシステム共通領域の 0000：0486H-0487H に書き込まれます．80286 等の CPU では違うデータが入っているようです．

CPU リセットとシステム・ポートのポート C・D_5/D_7を操作することで，プログラム中からも，プロセッサ ID を読み出すことができます(図 1-7)．

メモリ・マップ

PC98 シリーズは，8086 をベースにして拡張されてきました．CPU 自身は，8086 では 1M バイトを，80286 では 16M バイト(仮想アドレスは 1G バイト)，80386/80486 では 4G バイト(仮想アドレスは 64T バイト)のメモリを管理できるのですが，PC98 シリーズでは大まかに分けると 3 種類のメモリ・マップに分けられます．

〈図 1-5〉 CPU に対する命令

命　令	I/Oアドレス	R/W	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	機能
CPU Reset	00F0H	W	0	0	0	0	0	0	0	0	CPU，NDPの初期化．復旧できないエラーの対応，プロテクト・モードからリアル・モードへの移行に使用．
プロテクト・モード ON	00F2H	W	0	0	0	0	0	0	0	0	アドレス・バス上位4ビットのマスクを解除し100000H以上のメモリをアクセス可能にする．

〈図 1-6〉 プロテクト・モード OFF

命　令	I/Oアドレス	R/W	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	機能
プロテクト・モード・コントロール	00F6H	W	0	0	0	0	0	0	0	WAEN	WAENを1にすることにより，100000H以上のアドレスがアクセス不可になる．

◈ 基本のメモリ・マップ

基本のメモリ・マップは8086を搭載した初代のPC9801と同様の1Mバイトです．この1Mバイトのメモリ・マップも大きく分けると次の3種類の領域に分けられます．

(1) 00000H～9FFFFHまではRAMの空間
(2) A0000H～BFFFFHおよび
　E0000H～E7FFFHまではVRAM空間
(3) C0000H～DFFFFHまでは拡張用のROM空間
　E8000H～FFFFFHまではシステム用のROM空間

メモリ・マップを図1-8に示します．

システムROMとRAMの空間は各機種とも共通で，古い機種ではRAMが最初から640Kバイト分(00000H～9FFFFH)載っていない機種もありますが，拡張スロットにRAMを増設することで640Kバイトまで増やせます．また，RAMが最初から640Kバイト以上搭載されている機種では，ディップ・スイッチでバンク8～9(80000H～9FFFFH)の内蔵RAMを殺すことができます．これは，バンク切り替え式のRAMボードを拡張スロットに使用する場合に使います．

〈図 1-7〉 CPU の識別

プロセッサ名	コンポーネントID〈DHレジスタ〉	ステッピング/リビジョンID〈DLレジスタ〉
80386DX/SX	×3H	××H
80486DX	04H	0×H
80486DX50	04H	1×H
80486SX	04H	2×H
80487SX・MCP	04H	2×H
80486DX2	04H	3×H
80486DX・ODP	04H	3×H
80486SX・ODP	04H	3×H

◈ 拡張ROM空間

主にインターフェース等の拡張ボード用のROM空間で使われています．

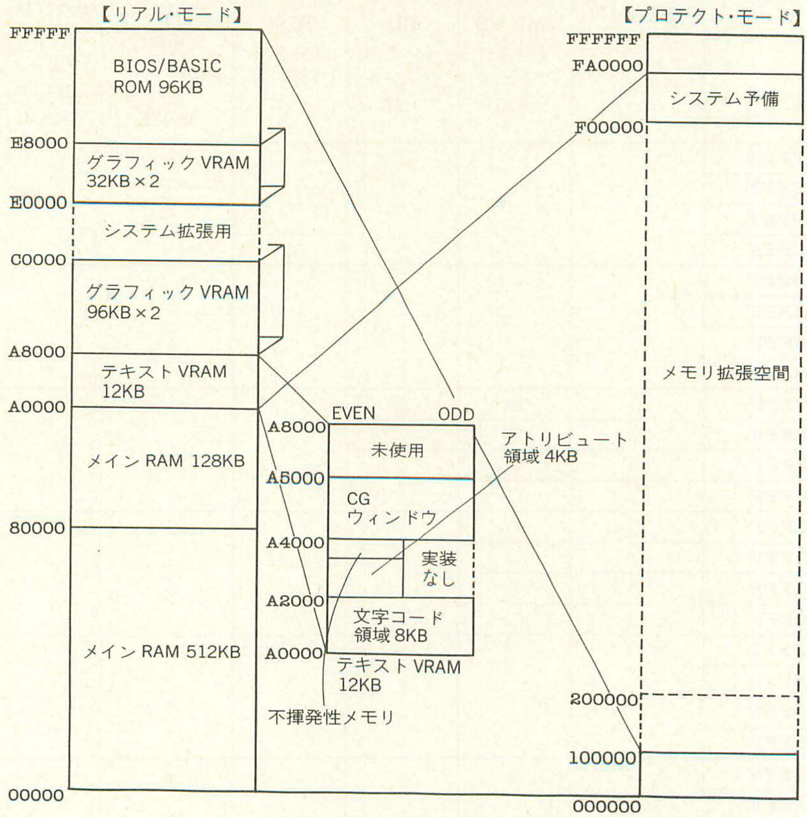
拡張ROM空間は，ユーザ用拡張ROM空間(C0000H～C7FFFH)とシステム用拡張ROM空間(C8000H～DFFFFH)に分かれます．システム用ROM空間は一般的にはメーカから供給されるインターフェース・ボード(FDDやHDD等)の基本的なボードのROM空間です．

拡張ボード用のROM空間は，図1-9に示すようにそのボードによって使用されるアドレスの標準的な位置が決まっています．中にはROMのアドレスを可変できるものもありますが，アドレスが固定されているものもあります．

また，拡張ボードではなく，本体自身でこの空間にROMを持ってる機種があります．代表的なものには，FM音源ボード用のROMや，HDDインターフェース用のROMがあります．さらに，ノート等には独自の仕様に基づいてシステム用のRAM空間や，レジューム用に使われている機種もあります．

拡張ボード用のROM空間に，RAMをマッピングしてEMSメモリとして使用されることもあります．CPUが80286以下の機種では，拡張スロット上のEMS対応RAMボードで，この領域にRAMをハードウェア的にマッピングして使用します．80386以上のCPUの機種では，CPU自身が持つメモリ・マネージメントの機能を使用して，プロテクト・メモリをこの領域にマッピングして使用します．

〈図 1-8〉基本メモリ・マップ(ノーマル・モード)



● 共通事項

・80000-9FFFFH は DIP-SW$_{3-6}$で RAM から切り離せる.

・A3FE2H, A3FE6H, A3FEAH, A3FEEH, A3FF2H, A3FF6H, A3FFAH, A3FFEH のメモリはバッテリ・バックアップされる. このうち, A3FE2H, A3FE6H, A3FEAH, A3FEEH, A3FF2H, A3FF6H がメモリ・スイッチとして使用される.

● EPSON 共通仕様

・DIP-SW$_{3-8}$が OFF(ノーマル・モード)のときは, 0 除算エラー例外(割り込みベクタ 00H), セグメント・オーバ・ラン(割り込みベクタ 0DH)に対応するアドレス(0:0000H-0003H, 0:0034H-0037H)はシステムで使用しており, ソフトウェアでライトしたデータと, リードしたデータが一致しない. このためワーク・エリアとして使用できない.

● 機種別仕様

・プロテクト・メモリは 286 以降の CPU でプロテクト・モード時のみアクセス可能である.

・0000:0400H-0000:05FFH はシステム共通域. システム共通域はソフトウェアからの参照のみを行い, 情報の更新は行わないこと.

・EMS 対応の機種では, B0000H-BFFFFH の 64KB の空間に EMS ページ・フレームの割り当てが可能.
 NEC:PC9801RA 以降で 80286CPU 以上の搭載機
 EPSON:286VX, 286VG, 386V, 386VR, 386S, 386G, 386GS, 386GE, 486GR, 486GF

・CG ウィンドウ(A400H-A4FFH)が搭載されていない機種.
 NEC:PC9801/E/F/M/U/VF/VM/UV/LV/CV/NV, PC98LT
 EPSON:286, 286V, 286VE, 286U, 286L

● NEC 機種別仕様

・1000000H 以降のメモリは PC9801Af でアクセス可能である.

・PC9801VX01, 21, 41 では高速グラフィック処理用に ROM を 2 組実装. 80286CPU/10MHz モード時に裏 ROM を使用する.

・GVRAM1~3, 同一アドレスに VRAM96KB を 2 組実装. PC9801U2 は一組.

・GVRAM4 は 16 色グラフィック対応 VRAM であり, 16 色未対応機では実装されていない. PC9801U2 では 32KB 一組.

〈図 1-9〉ROM 搭載オプション・ボードのアドレス空間

アドレス空間		640KB FD インターフェース −08 −09	1MB FD インターフェース −15	HD インターフェース −27	SCSI インターフェース −55	RS-232-C 拡張インターフェース 9861/K	サウンド・インターフェース U−03 −26/K	専用 HDD (IDE) および RAM ドライブ
システム予約	DF000〜DFFFF					○		
	DE000〜DEFFF							
	DD000〜DDFFF							
	DC000〜DCFFF				◎			
	DB000〜DBFFF					○		◎
	DA000〜DAFFF							
	D9000〜D9FFF							
	D8000〜D8FFF							
	D7000〜D7FFF	○	◎	◎		○		
	D6000〜D6FFF	◎	○					
	D5000〜D5FFF	○	○					
	D4000〜D4FFF	○	○					
	D3000〜D3FFF	○	○			◎		
	D2000〜D2FFF	○	○					
	D1000〜D1FFF	○	○					
	D0000〜D0FFF	○	○					
	CF000〜CFFFF					○		
	CE000〜CEFFF							
	CD000〜CDFFF						◎	
	CC000〜CCFFF							
	CB000〜CBFFF					○		
	CA000〜CAFFF							
	C9000〜C9FFF							
	C8000〜C8FFF							
ユーザ解放ROM空間	CF000〜CFFFF					○		
	C6000〜C6FFF							
	C5000〜C5FFF							
	C4000〜C4FFF							
	C3000〜C3FFF					○		
	C2000〜C2FFF							
	C1000〜C1FFF							
	C0000〜C0FFF							

◎：工場出荷時設定　○：変更可能アドレス

◆ VRAM空間

VRAM 空間は，テキスト VRAM とグラフィック VRAM の 2 種類があります．グラフィック VRAM は，古いディジタル RGB・8 色専用の機種では，A8000H〜BFFFFH の 96K バイトですが，アナログ RGB 対応・16 色の機種では，それに加えて，E0000H〜E8000H の 32K バイトが拡張されて，合計 128K バイトあります．

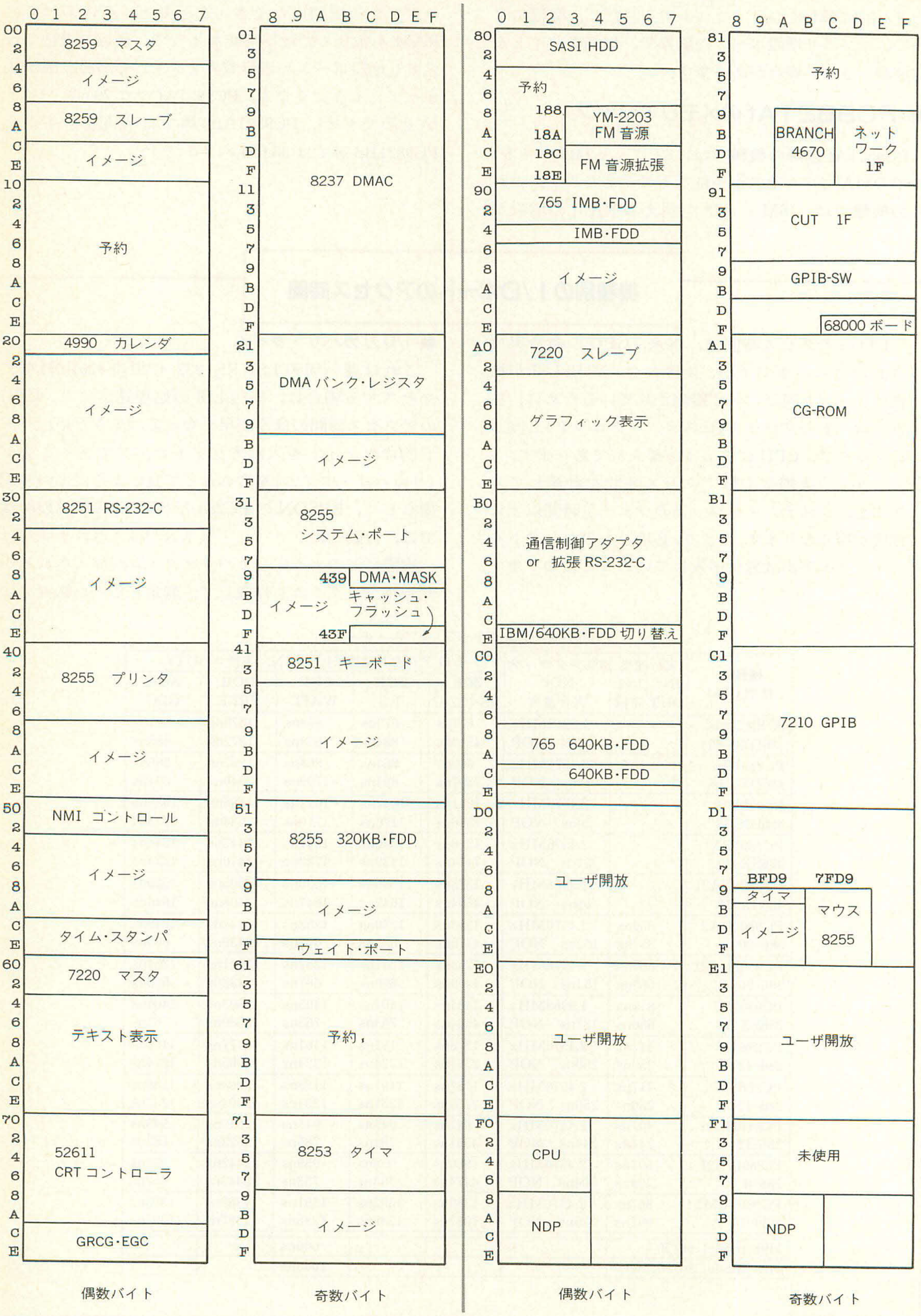
テキスト VRAM 空間は A0000H〜A3FFFH までですが，細かく分けると，文字データ用，アトリビュート用，不発揮メモリ，CG ウインドウ(A4000H〜A4FFFH)等に割り当てられています．

◆ 80286 以上のメモリ・マップ

CPU に 80286 以上を持つ機種の場合は，1M バイト以上のメモリを使用できます．ただし，MS-DOS を使用する場合は，80386 以上の CPU がないとプロテクト・メモリを有効に使用することができません．

メモリの上限は，DMA のアクセス範囲が 16M バイトのために，これに制限されています．また，メモリ空間上端の F00000H〜FFFFFFH(1M バイト)は，ROM 空間のイメージ等があり使用されないので，実際に使用できる RAM 空間は，640K バイト(コン

〈図 1-10〉 I/O ポート



8259 マスタ
イメージ
8259 スレーブ
イメージ
予約
4990 カレンダ
イメージ
8251 RS-232-C
イメージ
8255 プリンタ
イメージ
NMI コントロール
イメージ
タイム・スタンパ
7220 マスタ
テキスト表示
52611
CRT コントローラ
GRCG・EGC
偶数バイト
8237 DMAC
DMA バンク・レジスタ
イメージ
8255
システム・ポート
439 DMA・MASK
イメージ
キャッシュ・フラッシュ
43F
8251 キーボード
イメージ
8255 320KB・FDD
イメージ
ウェイト・ポート
予約
8253 タイマ
イメージ
奇数バイト
SASI HDD
予約
188
18A
18C
18E
YM-2203
FM 音源
FM 音源拡張
765 IMB・FDD
IMB・FDD
イメージ
7220 スレーブ
グラフィック表示
通信制御アダプタ
or 拡張 RS-232-C
IBM/640KB・FDD 切り替え
765 640KB・FDD
640KB・FDD
ユーザ開放
ユーザ開放
CPU
NDP
偶数バイト
予約
BRANCH
4670
ネットワーク
1F
CUT 1F
GPIB-SW
68000 ボード
CG-ROM
7210 GPIB
BFD9
7FD9
タイマ
イメージ
マウス
8255
ユーザ開放
未使用
NDP
奇数バイト

ベンショナル・メモリ)＋14M バイト(プロテクト・メモリ)の 14.6M バイトになります．ただし，機種によっては，メモリ増設ボードの限界で，14.6M バイトまで増設できない場合があります．

◆ PC9821Af のメモリ・マップ

PC9821Af 以降の機種では，32 ビット(4G バイト)まで DMA でアクセスが可能なものがあります．これらの機種では，16M バイトを超える RAM の増設が可能なものもあります．

アドレス空間はたくさんあっても，そのすべてに RAM を載せるのは不可能なようで，最近の機種ではメモリ増設ボードの増設限界でメモリの増設上限が決まってしまうようです．PC9821Af では 79.6M バイトが上限ですが，PC9821Ap2 等では 73.6M バイト，PC9821Bp 等では 35.6M バイトです．

機種別の I/O ポートのアクセス時間

I/O のアクセス時間は，本来，CPU の命令実行時間だけのはずですが、実際には，周辺 LSI が拡張スロットと同じバスに接続されているために，拡張スロットのクロック(システム・クロック)と同じになるまで，CPU にウェイトを入れてあります．

しかし，実際に I/O アクセス時間を計測してみますと，システム・クロックのウェイト時間以上に速度が遅くなります．つまり，必用以上のウェイトをハードウェアが故意に挿入していることになります．

● I/O リカバリ・タイム

これは遅い周辺 LSI(RS-232-C 用の 8251 等)をアクセスする場合に，周辺 LSI の処理速度より，CPU のアクセス時間のほうが早くなってしまうため，ハード的にウェイトを入れてソフトウェアでタイミング(リカバリ・タイム)を取らなくて良いようにという配慮らしく，EPSON の PC286/V/U/L/LE 以外の機種では，自動的にリカバリ・タイムが挿入されます．

実際にどのようにリカバリ・タイムが挿入されるか，測定用プログラムを作成して，測定してみた結果です．

〈図 1-A〉 デスクトップ(リアル・モード)

機種名 使用 CPU	実行速度 IN 上段 OUT 下段	8253・クロック NOP 実行速度	I/O アクセス時間(上段＝IN，下段＝OUT)				
			30H RS-232-C	5CH T.S	5FH WAIT	DOH FREE	AOH GDC
PC-H98 改 m105 486DX2-80		1.9968MHz 15ns NOP	877ns 4513ns	879ns 884ns	878ns 858ns	1375ns 1372ns	881ns 885ns
PC9801Ap 486DX2-66		2.4576MHz 19ns NOP	863ns 7397ns	864ns 894ns	864ns 1709ns	1897ns 2540ns	866ns 894ns
PC9801As 486DX-33		2.4576MHz 30ns NOP	1041ns 7396ns	1042ns 1126ns	1042ns 1710ns	1898ns 2534ns	1039ns 1125ns
PC9801BX 486SX-20		2.4576MHz 51ns NOP	1346ns 7414ns	1346ns 1439ns	1347ns 1728ns	2445ns 2440ns	1348ns 1434ns
PC9801FA21 486SX-16		1.9968MHz 65ns NOP	1528ns 4504ns	1526ns 1647ns	1530ns 1647ns	2505ns 2506ns	1529ns 1648ns
PC9801DA2 386-20	659ns 557ns	2.4576MHz 152ns NOP	1253ns 4470ns	1250ns 665ns	1252ns 666ns	2441ns 2439ns	1249ns 665ns
PC9801RA21 386-20	667ns 565ns	2.4576MHz 154ns NOP	1253ns 4469ns	1251ns 666ns	1252ns 667ns	2441ns 2438ns	1251ns 666ns
PC9801ES 386SX-16	810ns 686ns	1.9968MHz 187ns NOP	1404ns 4504ns	1404ns 763ns	1405ns 763ns	2503ns 2500ns	1405ns 763ns
PC286VF 286-12	417ns 250ns	2.4576MHz 250ns NOP	1178ns 2751ns	1181ns 1224ns	1181ns 1224ns	4177ns 3005ns	1180ns 1224ns
PC286VE 286-12	417ns 250ns	2.4576MHz 250ns NOP	1187ns 2755ns	1191ns 1231ns	1188ns 1234ns	3004ns 3004ns	1186ns 1232ns
PC9801RX2 286-12	407ns 244ns	2.4576MHz 244ns NOP	947ns 4551ns	945ns 784ns	945ns 785ns	2519ns 2523ns	945ns 782ns
PC9801VX21 286-8	507ns 304ns	2.4576MHz 304ns NOP	960ns 4474ns	958ns 753ns	959ns 753ns	2442ns 2443ns	957ns 757ns
PC9801VM2 V30-10	867ns 867ns	2.4576MHz 325ns NOP	1389ns 1387ns	1392ns 1388ns	1391ns 1388ns	1389ns 1387ns	1393ns 1391ns
H98 model 60/70					600ns		
H98 model 80/90/100					1000ns		

I/O ポート

PC98 シリーズで使用されている，周辺インターフェース LSI のデータ・バスの幅は 8 ビットのものです．拡張スロットのバス幅は 16 ビットですから，データ・バスを上位 8 ビットと下位 8 ビットの二つに分けて使用します．このため，上位につながれた LSI は奇数アドレスになり，下位につながれた LSI は偶数アドレスに割り当てられます．したがって，LSI 自身が複数のアドレスを使用する場合は，CPU からは奇数・偶数の一つ飛びのアドレスに見えます(図 1-10)．

80386DX や 80486 等の CPU は 32 ビットのバスを持っています．しかし，拡張スロットのバス幅は 16 ビットですから，32 ビットのバスを上位 16 ビットと下位 16 ビットに分けて，16 ビット・バスとして使用する必要があります．これらの CPU ではバス・サイ

プログラムは，NOP，IN，OUT の命令を実行する時間を計測するものです．

この結果，RS-232-C やキーボード・インターフェース用の 8251 には，どの機種にもかなりのウェイトが挿入されているようです．D0H からのユーザに解放されている I/O アドレスも，他のアドレスと比べて余計にウェイトがかかる傾向があります．また，PC9801FA 以降の機種は，全体的にウェイトが多くなっています．

H98 シリーズや 80486 以上の CPU 以上では標準となった OUT　5FH，AL 命令も，機種によって結構時間が変わることもわかりました．

● 参考

対象となる命令の実行時間(クロック数)．

命令	CPU	クロック数		
		8086	80286	80386
IN　AL，DX		8	5	13(27)
OUT　DX，AL		8	3	11(25)
NOP		3	3	3

カッコ内は仮想 86 モード

〈図 1-B〉ノート関連

機種名 使用 CPU	8253・クロック NOP 実行速度	I/O アクセス時間(上段＝IN，下段＝OUT)				
		30H RS-232-C	5CH T.S	5FH WAIT	D0H FREE	A0H GDC
PC9801NS/E 改 386SX-20	2.4576MHz 150ns　NOP	1251ns 3656ns	1251ns 18060ns	1253ns 744ns	2033ns 2036ns	1250ns 744ns
PC9801NS/T 386SL-20	1.9968MHz 151ns　NOP	1581ns 4628ns	1581ns 993ns	1581ns 1002ns	2629ns 2630ns	1582ns 983ns
PC9801NS 16MHz 386SX-16	2.4576MHz 189ns　NOP	2507ns 4371ns	3144ns 2636ns	3145ns 2637ns	2499ns 2376ns	2503ns 1979ns
PC9801NV V30-16	1.9968MHz 195ns　NOP	1120ns 4604ns	1123ns 4602ns	1124ns 1121ns	2504ns 2503ns	1120ns 1120ns
PC98LT　V50-8MHz V50-8	1.9968MHz 374ns　NOP	1443ns 1441ns	1443ns 1441ns	1443ns 1442ns	1444ns 1442ns	1443ns 1441ns

〈図 1-C〉サイリックス CPU に載せ替えたもの

機種名 使用 CPU	8253・クロック NOP 実行速度	I/O アクセス時間(上段＝IN，下段＝OUT)				
		30H RS-232-C	5CH T.S	5FH WAIT	D0H FREE	A0H GDC
PC386M 仮想 86 CX486SLC2-16M MELEMM.386Ver5.19	2.4576MHz 89ns　NOP	1500ns 2749ns	1598ns 1702ns	1601ns 1703ns	3505ns 3499ns	1503ns 1600ns
PC386S 仮想 86 MELEMM.386Ver5.22 Cx486DLC25(HSB.EXE CXP)	2.4576MHz 115ns　NOP	1728ns 2238ns	1620ns 892ns	1619ns 918ns	2830ns 2664ns	1615ns 918ns
PC386S　Cx486DLC25 (HSB.EXE　CXP)	2.4576MHz 118ns　NOP	1595ns 2237ns	1411ns 889ns	1412ns 913ns	2774ns 2660ns	1409ns 912ns
PC9801NS 20MHz＋486SLC	2.4576MHz 149ns　NOP	1356ns 4308ns	1356ns 918ns	1356ns 920ns	2274ns 2278ns	1356ns 874ns
PC286UX＋486SLC 24MHz real mode	2.4576MHz 149ns　NOP	1500ns 2746ns	1502ns 1252ns	1501ns 1252ns	2341ns 1801ns	1501ns 1250ns

〈図 1-11〉ウェイトの入れ方

CPU から周辺 LSI に対して連続アクセスを行う場合，周辺 LSI の動作が完了するまで待ってから，次のアクセスを行わなくてはならない．そのために，8086，V30 を CPU に持つ機種では「NOP」を，80286，80386 を CPU に持つ機種では「JMP ＿$+2」を，80486，Pentium を CPU に持つ機種では「OUT ＿ 5Fh，AL」等の命令を挿入してタイミングを取る必要がある．

8086　　[NOP] 使用(3clock)

LSI		リカバリ・タイム (ns)	WR-WR 5/8	RD-RD 5/8	WR-RD 5/8	RD-WR 5/8
8237	DMAC	400	0 0	0 0	0 1	0 0
8253	タイマ	1000	0 1	0 1	1 2	0 1
8255	PIO	850	0 1	0 1	1 2	0 1
8259	PIC		0 0	0 0	1 1	0 0
8251	モード初期化	$6t_{cy}$	3 6			
SIO	非同期モード	$8t_{cy}$	4 8			
	同期モード	$16t_{cy}$	8 16			
765	FDC		0 0	0 0	0 0	0 0
7220	グラフ	標準 2516	0 3	0 3	0 3	0 3
GDC	2.5MHz	高解像 1710	0 2	0 2	0 2	0 2
7210	GPIB	250	0 0	0 0	0 1	0 0

70116(V30)　　[NOP] 使用(3clock)

LSI		リカバリ・タイム (ns)	WR-WR 8/10/16	RD-RD 8/10/16	WR-RD 8/10/16	RD-WR 8/10/16
8237	DMAC	400	0 0 0	0 0 0	1 1 1	0 0 0
8253	タイマ	1000	1 2 3	1 2 3	2 3 4	0 0 2
8255	PIO	850	1 1 2	1 1 2	2 3 4	0 0 1
8259	PIC		0 0 0	0 0 0	0 0 0	0 0 0
8251	モード初期化	$6t_{cy}$	6 6 13			
SIO	非同期モード	$8t_{cy}$	9 9 19			
	同期モード	$16t_{cy}$	20 20 40			
765	FDC		0 0 0	0 0 0	0 0 0	0 0 0
7220	グラフ	標準 2516	4 5 -	4 5 -	5 6 -	3 4 -
	2.5MHz	高解像 1710	2 2 3	2 2 3	3 3 5	1 2 2
GDC	グラフ/テキスト	標準 1260	0 1 -	0 1 -	1 2 -	0 0 -
	5MHz	高解像 855	0 0 0	0 0 0	1 1 1	0 0 0
7201	通信制御	300	0 0 0	0 0 0	1 1 1	0 0 0
7210	GPIB	250	0 0 0	0 0 0	1 1 1	0 0 0

＊ PC9801NL を除く

80286　　[JAP $+2] 使用(7clock)

LSI		リカバリ・タイム (ns)	WR-WR 8/10/12	RD-RD 8/10/12	WR-RD 8/10/12	RD-WR 8/10/12
8237	DMAC	400	1 1 1	0 0 0	1 1 1	0 0 0
8253	タイマ	1000	1 1 2	1 1 2	1 1 2	1 1 2
8255	PIO	850	1 1 1	1 1 1	1 1 1	1 1 1
8259	PIC		1 0 0	0 0 0	1 1 1	0 0 0
8251	モード初期化	$6t_{cy}$	3 3 3			
SIO	非同期モード	$8t_{cy}$	4 4 4			
	同期モード	$16t_{cy}$	7 7 7			
765	FDC		0 0 0	0 0 0	0 0 0	0 0 0
7220	グラフ	標準 2516	2 3 3	2 2 2	2 3 3	2 2 2
	2.5MHz	高解像 1710	2 2 2	1 1 1	2 2 2	1 1 1
GDC	グラフ/テキスト	標準 1260	1 1 1	1 1 1	1 1 2	1 1 1
	5MHz	高解像 855	1 1 1	0 0 0	1 1 1	0 0 0
7201	通信制御	300	1 1 1	0 0 0	1 1 1	0 0 0
7210	GPIB	250	1 1 1	0 0 0	1 1 1	0 0 0

・EPSON の機種で，PC286，PC286V，PC286U，PC286L，PC286LE 以段の機種と，NEC の PC98DO＋は，周辺 LSI への連続アクセスのためのリカバリ・タイムをハードウェアで生成するために，プログラムでのタイミングを取る必要はない．
・80486 以上の CPU を持つ機種では，ポート 5FH をライトすることで一定時間のウェイトを確保することができる．EPSON の機種は約 700ns(最低 500ns)のウェイトがかかる．

〈図 1-11〉ウェイトの入れ方(つづき)

80386　　[JAP $+2] 使用(7+*m*clock)

LSI		リカバリ・タイム (ns)	WR-WR 12/16/20	RD-RD 12/16/20	WR-RD 12/16/20	RD-WR 12/16/20	
8237	DMAC	400	1 1 1	0 0 0	1 1 1	0 0 0	
8253	タイマ	1000	2 2 2	1 1*1	2 2 2	1 1*1	*1=2
8255	PIO	850	2 2 2	1 1 1	2 2 2	1 1 1	
8259	PIC		0 0 0	0 0 0	1 1 1	0 0 0	
8251	モード初期化	$6t_{cy}$	6 6 6				
SIO	非同期モード	$8t_{cy}$	8 8 8				
	同期モード	$16t_{cy}$	16 16 16				
765	FDC		0 0 0	0 0 0	0 0 0	0 0 0	
7220	グラフ	標準 2516	- - 4	- - 4	- - 4	- - 4	
	2.5MHz	高解像 1710	3 3 3	2 2*2	3 3 3	2 2*2	*2=3
GDC	グラフ/テキスト	標準 1260	- - 2	- - 1	- - 2	- - 1	
	5MHz	高解像 855	1 1 1	0*3*3	1 1 1	0*3*3	*3=0
	テキスト		1 1 1	2*4*5	1 1 1	0*6*7	*x=0
7201	通信制御	300	1 1 1	0 0 0	1 1 1	0 0 0	
7210	GPIB	250	1 1 1	0 0 0	1 1 1	0 0 0	

*　PC9801FX，PC9801FS，PC9821model S1，S2 は除く

*1　PC9801NS/T，PC9801US は 1，それ以外は 2

*2　PC9801NS/T，PC9801US は 2，それ以外は 3

*3　PC9801DA は 1，それ以外は 0

*4　PC9801DA は 1，PC9801RS21，51，PC9801NS/T，C9801NS/L は 2，それ以外は 0

*5　PC9801DA は 1，PC9801T，PC9801NS/L は 2，それ以外は 0

*6　PC9801DA は 1，PC9801NS/T は 2，それ以外は 0

*7　PC9801DA は 1，PC9801T は 2，それ以外は 0

80386 PC9801FX, PC9801FS, PC9821model S1, S2　　[JAP $+2] 使用(7+*m* clock)

LSI		リカバリ・タイム (ns)	WR-WR 12/20	RD-RD 12/20	WR-RD 12/20	RD-WR 12/20
71037	DMAC	125	0 0	0 0	0 0	0 0
71053	タイマ	200	0 0	0 0	0 0	0 0
71055	PIO	200	0 0	0 0	0 0	0 0
71059	PIC	250	0 0	0 0	0 0	0 0
71051	モード初期化	$6t_{cy}$	9 14			
SIO	非同期モード	$8t_{cy}$	9 14			
	同期モード	$16t_{cy}$	9 14			
72065	FDC		0 0	0 0	0 0	0 0
72020	グラフ	標準 2516	4 5	4 5	4 5	4 5
	2.5MHz	高解像 1710	2 4	2 4	2 4	2 4
GDC	グラフ/テキスト	標準 1260	2 3	2 3	2 3	2 3
	5MHz	高解像 855	1 2	1 2	1 2	1 2
7201	通信制御	300	0 0	0 0	0 0	0 0
7210	GPIB	250	0 0	0 0	0 0	0 0

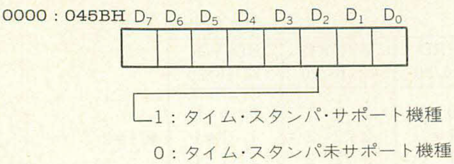
80486，Pentium　　[OUT 5FH，AL] 使用

LSI		リカバリ・タイム (ns)	WR-WR FA/BA/ex	RD-RD FA/BA/ex	WR-RD FA/BA/ex	RD-WR FA/BA/ex
71037	DMAC	125	0 0 0	0 0 0	0 0 0	0 0 0
71053	タイマ	200	0 0 0	0 0 0	0 0 0	0 0 0
71055	PIO	200	0 0 0	0 0 0	0 0 0	0 0 0
71059	PIC	250	0 0 0	0 0 0	0 0 0	0 0 0
71051	モード初期化	$6t_{cy}$	6 2 5			
SIO	非同期モード	$8t_{cy}$	6 2 5			
	同期モード	$16t_{cy}$	6 4 5			
72065	FDC		0 0 0	0 0 0	0 0 0	0 0 0
72020	グラフ	標準 2516	2 - -	2 - -	2 - -	2 - -
	2.5MHz	高解像 1710	1 0 1	1 0 1	1 0 1	1 0 1
GDC	グラフ/テキスト	標準 1260	1 - -	1 - -	1 - -	1 - -
	5MHz	高解像 855	1 0 1	1 0 1	1 0 1	1 0 1
7201	通信制御	300	0 0 1	0 0 0	0 0 1	0 0 0
7210	GPIB	250	0 0 1	0 0 0	0 0 1	0 0 0

FA　PC9801FA　　BA　PC9801BA，BX

ex　PC9821Ap，As，Ae，Ce，PC9821Ne，PC9801NA，NS/R，NX/C，P，Af

〈図 1-12〉 OUT 5Fh,AL サポート機種識別ビット



ジング機能があり，32 ビットのバスのうち同時に 16 ビットしか使わなくできますので，この機能を使用します．

PC9801/E/F/M 等の初期の PC98 シリーズでは，各 I/O デバイスのアドレス・デコードが，下位 8 ビット分しか行われていませんでした．そのため，8086 自身は 16 ビット(65536 個)のアドレスを持つにもかかわらず，8 ビット(256 個)分のアドレスしか使用できません．

PC9801VF/VM 以降の機種では一部改められて，本体内部の I/O デバイスが A_{11}，A_{12} もデコードされるようになりましたので，I/O アドレス空間が 4 倍になったのですが，互換性のためか，拡張された空間はユーザには開放されず，その機種固有の非公開の I/O ポートが割り当てられているようです．

ユーザに開放されている I/O ポート・アドレスは，×nDOH〜×nEFH であり，n は 0〜7 までです．n=8〜F までは NEC のリザーブになっています．7FD9H〜7FDFH(奇数)までにはタイマ LSI や，マウス関連に I/O がありますので，下位 8 ビット・デコードのみのボードは使用できません．

8 ビット・デコードでユーザに解放されている I/O アドレスを使うと，D9H，DBH，DDH，DFH を除いた DOH〜EFH の 28 バイトしかありません．他にもこのアドレスを使っている拡張基板は多く，他の基板と併用して使うと，さらに使えるアドレスは減ってしまいます．

厳密には×nDOH〜×nEFH(n=8〜F)は NEC のリザーブですから，今後，拡張基板を設計する場合は最低でも 12 ビット・デコードする必要があると思われます．12〜16 ビット・デコードすることで，使用できるアドレスが格段に増えます．しかし，フルアドレス・デコードをしても，連続したアドレス空間は最大 32 バイトしか取れません．8 ビット・データ・バスを持つデバイスでは，上位/下位バイトのいずれかしか使えませんので，最大 16 バイトしか連続アドレスが取れません．レジスタをたくさん持つ，高機能な LSI を使用する場合に問題が出そうです．

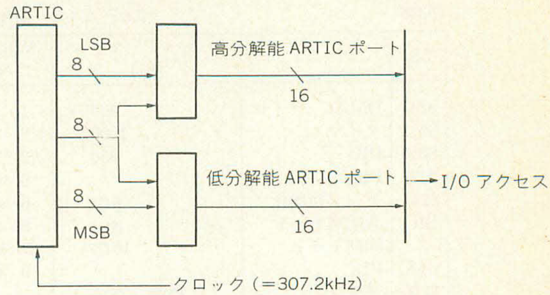
◈ I/O の連続アクセスの制限

CPU から周辺 LSI へ連続したアクセスを行う場合，LSI の処理速度より CPU のアクセスのほうが速くなる場合があります．最初に CPU から与えられた命令の処理が終了する前に，次の CPU からの命令が来てしまい，LSI が正常な動作をしない場合があります．これを保証するために，周辺 LSI へ連続したアクセスをする場合は，NOP，JMP 命令等でウェイト(リカバリ・タイム)を入れる必要があります(図 1-11)．

8086/V30 系列の CPU では NOP(3 クロック)を，80286/80386 の CPU では JMP　$+2(7 クロック)(次の命令に JUMP する)を，80486/Pentium では OUT　5FH，AL をいくつか実行することでウェイトを入れます．

CPU に 80486 を持つ機種や PC-H98 シリーズでは，リカバリ・タイムを CPU の実行速度に依存せずに作るために，OUT　5FH，AL という命令を使います．これは I/O ポートの 5FH 番地に書き込みをするだけで，ハードウェア的にウェイトを取る機能で，機種によってかなりばらつきがありますが，最低でも 600 ns

〈図 1-13〉 タイム・スタンパの構成



高分解能 ARTIC ポート：ARTICの下位16 ビットを読み出すポート
低分解能 ARTIC ポート：ARTICの上位16 ビットを読み出すポート

(a) タイム・スタンパの構成

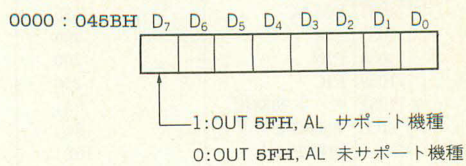
二つの I/O ポートは，設定する時間によって使い分ける

ポート名	分解能	最大値(*1)	アドレス
高分解能 ARTIC ポート	3.26μs	213.6ms	005CH (ワード)
低分解能 ARTIC ポート	834.6μs	54.7s	005EH (ワード)

(*1) 最上位ビットの変化する周期

(b) ARTIC ポート

〈図 1-14〉 タイム・スタンパ・サポート機種識別ビット



〈図1-15〉ディップ・スイッチSW

参照ポートはプログラム中から読み出すことができるポート

SW_1	ON	OFF	参照ポート
1	専用高解像度ディスプレイ	標準ディスプレイ	0033H(D3)
2	スーパ・インポーズを使用する	使用しない	
3	プラズマ・ディスプレイを使用する	使用しない	0042H(D4)
4	内蔵FDD番号，#3，#4	#1，#2	
5	RS-232-Cモード選択(注1)		7FDDH(D0)
6	RS-232-Cモード選択(注1)		7FDDH(D1)
7			
8	拡張グラフィック・モード	基本グラフィック・モード	0042H(D3)

SW_2	ON	OFF	参照ポート
1			0031H(D0)
2	ターミナル・モード	BASICモード	0031H(D1)
3	80字/行	40字/行	0031H(D2)
4	25行/画面	20行/画面	0031H(D3)
5	メモリ・スイッチ保持	メモリ・スイッチ初期化	0031H(D4)
6	内蔵HDDを切り離す	内蔵HDDを使用する	0031H(D5)
7	FDモータ制御あり(注2)	なし	0031H(D6)
8	GDC5MHzモード	2.5MHz	0031H(D7)

SW_3	ON	OFF	参照ポート
1	内蔵FD固定モード	内蔵FDプログラム・モード	00BEH(D2)
2	内蔵FD・640Kモード	内蔵FD・1MBモード	00BEH(D3)
3	内蔵HD・DMA・#1/#2(注3)　#0		
4			
5	(注4)		
6	RAM512K	RAM640K	7FDBH(D6)
7	メモリ・アクセス　0WAIT(注5)	1WAIT	
8	動作CPU・286/386/486	V30	0042H(D1)/7FDDH(D2)

(注1)RS-232-Cモード選択

5	6	同期モード
ON	ON	BCI同期
ON	OFF	ST2同期
OFF	ON	同期刻時機構
OFF	OFF	調歩同期

(注2)PC9801LV/CV/UVのみ

(注3)PC98RL/PC9801T/DX/DS/DA/CS/FX/FS/FA/PC9821Ae/As/Ap/Afのみ

(注4)機種によって機能が変わる

(注5)NECでは未定義

●スーパ・インポーズ機能は，PC9801/E/F/M以外の機種でディジタルRGB出力を持つ機種のみで使用可能(オプション)．RGBコネクタの「DOT CLOCK」が入力端子になる．

●プラズマ・ディスプレイは，PC9801/E/F/M以外の機種でディジタルRGBを持つ機種のみで使用可能(オプション)．CLOCKとのスキュを保証したRGB信号を出力する．

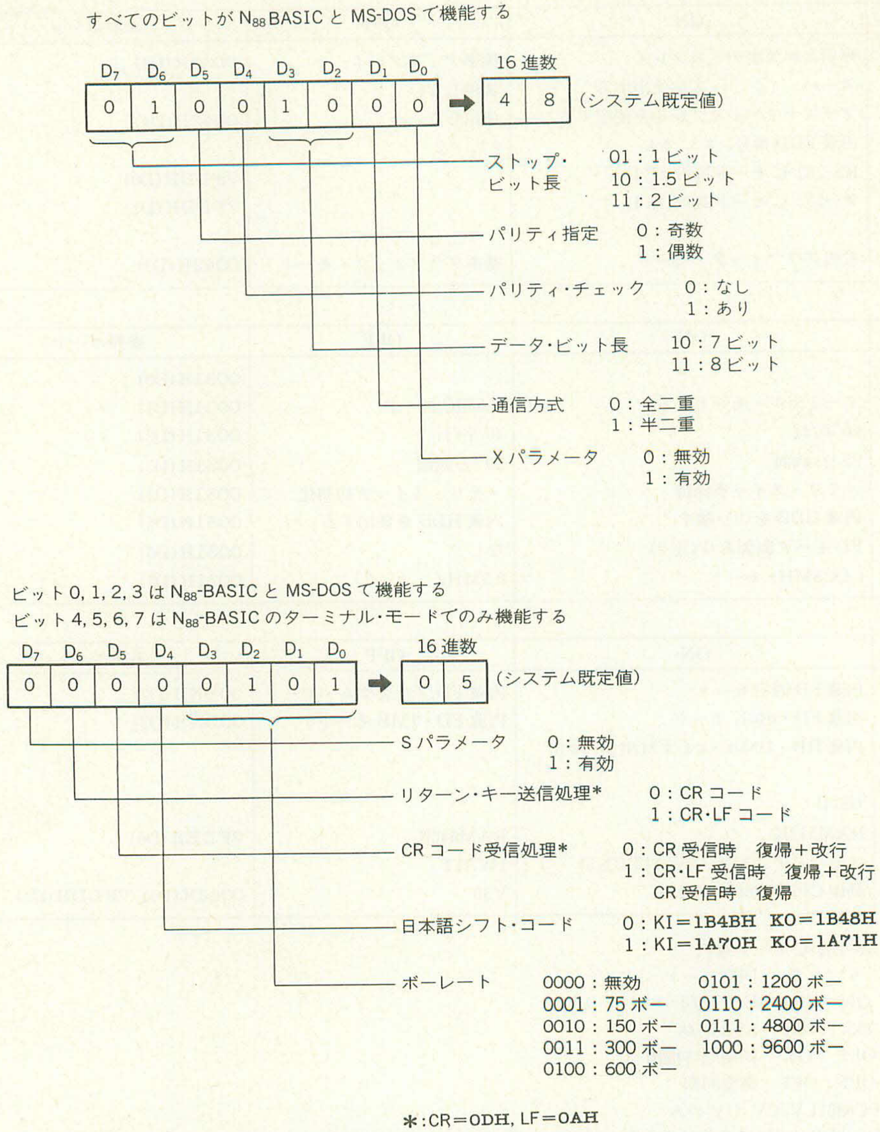
程度のリカバリ・タイムが得られます．この機能を持つ機種を判別するには，システム共通領域の0000：045BH・D7が“1”であれば，サポートされます(図1-12)．

また，機種によってはハードウェアで自動的にある程度のリカバリ・タイムを取ってくれるものもありますが，不十分な場合もありますし，他機種での動作保証のためにも，指定された方法でリカバリ・タイムを取るほうが良いと思われます(p.10コラム参照)．

◈ タイム・スタンパ

I/Oリカバリ・タイムを保証する手段として，タイム・スタンパを搭載している機種もあります．タイム・スタンパは，クロック307.2 kHzで動作する24ビット・バイナリ・カウンタで，CPU等に依存されずに動作します．カウンタに対して，初期設定や数値設定

〈図 1-16〉メモリ・スイッチ



はできず，フリーランしているのでプログラム中で何度か読み出して，必要な時間が経過したか調べる必要があります．

タイム・スタンパの構成を図 1-13 に示します．

タイム・スタンパは PC-H98 や，CPU に 80486 を採用した機種に搭載されているようで，サポート機種の判別は，図 1-14 のシステム共通領域で識別します．

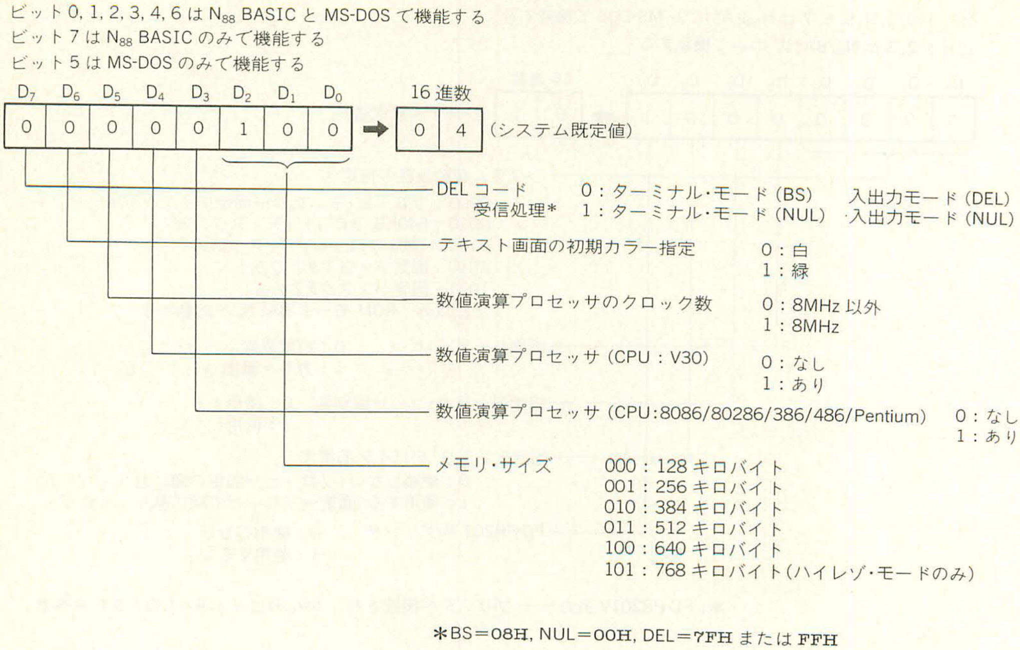
◈ ディップ・スイッチ

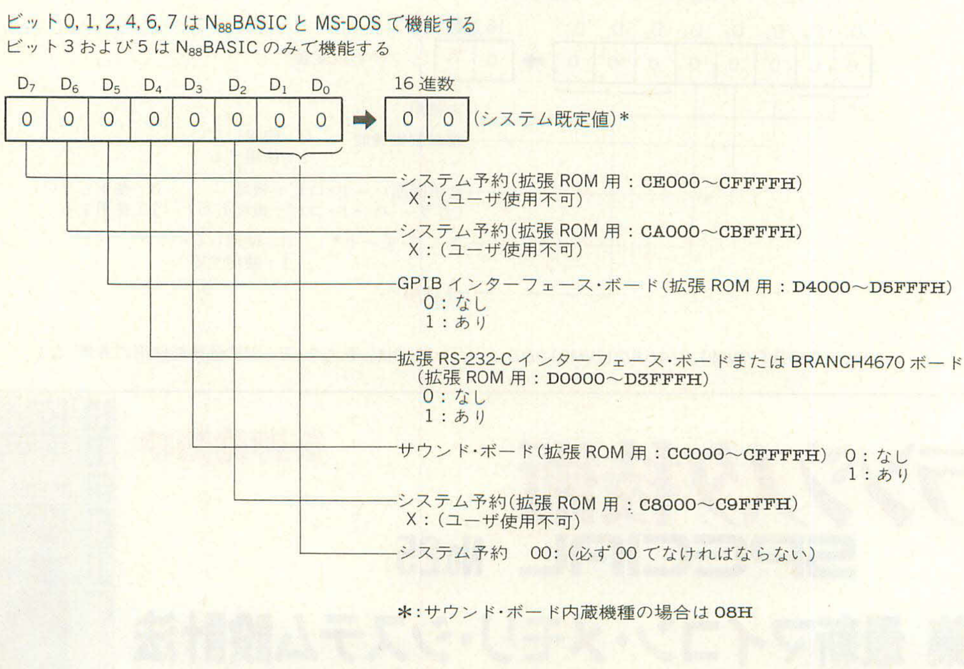
ディップ・スイッチは，機種によってその意味が違っていたり，ハードウェアのスイッチではなくメニュ・プログラムから設定する機種もあります．メニュ・プログラムは，「HELP キー」を押しながらリセットすれば起動します．

図 1-15 の表は一般的なディップ・スイッチ表で，一部意味が違う機種もあります．一部または全部がソフトウェア・ディップ・スイッチになっているものや，スイッチの数，順番が違うものもあるので，詳細はそれぞれの機種のマニュアルを参照する必要があるでしょう．

図 1-15 にディップ・スイッチ SW の割り付けを示します．

〈図 1-16〉 メモリ・スイッチ(つづき)





◈ メモリ・スイッチ

メモリ・スイッチは，各種システムの設定値を不発揮メモリで保存します．RS-232-C パラメータ，メモリ・サイズ，起動ドライブ指定等の機能があります．

メモリ・スイッチは通常のメモリ空間にあるために，不用意に書き換えないように，ライト・プロテクトをかけることができます．不揮発メモリへの書き込み許可・禁止は I/O アドレス(6AH)のモード・レジスタで行います(図 1-16)．

〈図 1-16〉 メモリ・スイッチ(つづき)

ビット 0, 1, 4, 5, 6, 7 は N_{88}BASIC と MS-DOS で機能する
ビット 2, 3 は N_{88} BASIC のみで機能する

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	1

➡ 16 進数 0 1 (システム既定値)

- D_7～D_4：システム起動装置の指定
  - 0000：フロッピ・ディスク→固定デイスクの順にサーチ
  - 0010：640KB フロッピ・ディスクのみ
  - 0100：1MB フロッピ・ディスクのみ
  - 1010：固定ディスク#1のみ
  - 1011：固定ディスク#2のみ
  - 上記以外：ROM モード BASIC が起動する
- D_3：画面ハード・コピー　0：白黒画面ハード・コピー　1：カラー画面ハード・コピー*
- D_2：固定ディスク・ユーザ識別名　0：使用する　1：使用しない
- D_1：固定ディスク・デバイス名優先
  - 0：使用しない(フロッピ→固定の順に割りふられる)
  - 1：使用する(固定→フロッピの順に割りふられる)
- D_0：PC-PR201 系プリンタ　0：使用しない　1：使用する

＊：PC-PR201V 系カラー・プリンタが接続され，SW_6 のビット 4=1 のときのみ有効

すべてのビットが N_{88} BASIC のみで機能する

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0

➡ 16 進数 0 0 (システム既定値)

- D_7, D_6：未使用
- D_5：電話制御機能　0：使用しない　1：使用する
- D_4：拡張画面ハード・コピー機能(カラー・ハード・コピー機能も可)　0：使用しない　1：使用する
- D_3：モニタ・モード*　0：使用しない　1：使用する
- D_2～D_0：未使用

＊：PC9801VF2/VM0, 2, 4/UV21/VM21/VX0, 2, 4/UV21では，モニタ・モード拡張機能使用の有無となる

トランジスタ技術 SPECIAL No.25

好評発売中

特集 最新マイコン・メモリ・システム設計法

DRAM,SRAMの動作からデュアルポートRAM,FIFOの活用まで



目　次

EPROM活用の基礎技術
SRAM活用の基礎技術
疑似SRAM活用の基礎技術
DRAM活用の基礎技術
デュアルポートRAM活用の基礎技術
FIFOメモリ活用の基礎技術
タイミング計算の方法
8ビットMPUのためのメモリ・システム
16ビットMPUのためのメモリ・システム
32ビットMPUのためのメモリ・システム

B5判 160頁
定価1,540円(税込)

CQ出版社

PC9801 シリーズの割り込み

8086 系の CPU では，リセットを除く割り込みのすべてがベクタ(割り込み先アドレス)が，メモリ空間の先頭の 0000：0000H～03FFFH までの 1K バイトの領域に割り当てられています．

ベクタは全部で 256 個あり，CPU の内部処理割り込みやハードウェア割り込み，ソフトウェア割り込み等が割り当てられています．

割り込みベクター覧を図 1-17 に示します．

一つのベクタは 4 バイトあり，セグメント部 2 バイトとオフセット部 2 バイトで成っています．割り込みがかかった場合は，このアドレスへ FAR JUMP します．

例えば，割り込み 3 番が発生すると，3×4=12 で，000CH-000DH のデータをオフセットに，000EH-000FH のデータをセグメントに入れて，FAR JUMP します．

このデータは，小さい番地のアドレス(000CH)に

〈図 1-17〉割り込みベクタ

ベクタ・アドレス	ベクタ番号	用途
0—3	0	除算エラー
4—7	1	シングル・ステップ
8—B	2	NMI
C—F	3	INT_3
10—13	4	オーバ・フロー
14—17	5	コピー(COPY)キー
18—1B	6	STOP キー
1C—1F	7	インターバル・タイマ
20—23	8	タイマ
24—27	9	キーボード
28—2B	A	CRTV(V-SYNC)
2C—2F	B	拡張バス INT_0
30—33	C	RS-232-C(ch0)
34—37	D	拡張バス INT_1
38—3B	E	拡張バス INT_2(CMT)(注 1)
3C—3F	F	システム予約

ベクタ・アドレス	ベクタ番号	用途
40—43	10	プリンタ/NDP(注 2)
44—47	11	拡張バス INT_3(HD)
48—4B	12	拡張バス INT_{41}(640KBFD)
4C—4F	13	拡張バス INT_{42}(1MBFD)
50—53	14	拡張バス INT_5(サウンド)
54—57	15	拡張バス INT_6(マウス)(注 3)
58—5B	16	NDP/プリンタ(注 4)
5C—5F	17	ノイズ(システム予約)(注 6)
60—63	18	KB/CRT BIOS
64—67	19	RS-232-C BIOS
68—6B	1A	プリンタ BIOS
6C—6F	1B	DISK BIOS
70—73	1C	カレンダ BIOS
74—77	1D	システム予約
78—7B	1E	N_{88} BASIC
7C—7F	1F	システム予約
80～FF	20～3F	システム予約
100～1FF	40～7F	ユーザ用

使用されていないベクタには，ダミー処理のアドレスが設定されている．
ベクタ番号 8～17 はハードウェア割り込みに使用．
注 1：PC9801 のみ INT_2は CMT に使用されている．
　　ハイレゾ・モード時，マウスは INT_2に固定されている．
注 2：8086，70116CPU 動作時プリンタ，80286，386，486，Pentium 動作時 NDP.
注 3：ノーマル・モード時，マウスの割り込みはストラップ・スイッチにより INT_0～INT_6に変更可能．
　　ハイレゾ・モード時，マウスは INT_2に固定されている．
注 4：ノーマル・モードでは 8086，70116CPU 動作時 NDP，80286，386，486，Pentium 動作時 NC.
　　ハイレゾ・モード時プリンタ．
注 5：ハイレゾ・モード時のみ GRAPH BIOS.
注 6：PC9821Ap，As，Ae，Af，Ne はシステム・タイマに使用．

〈リスト 1-1〉割り込みベクタのリスト

```
0000:0000  54 53 F5 09 36 09 80 FD F0 08 80 FD 36 09 80 FD
0000:0010  36 09 80 FD F2 77 60 00 2D 77 60 00 36 09 80 FD
0000:0020  03 06 80 FD 17 09 73 1B 37 09 80 FD 37 09 80 FD
0000:0030  33 18 80 FD 1E 00 00 D4 37 09 80 FD 37 09 80 FD
```

は下位バイトが，大きい番地のアドレス(000DH)には上位バイトが入っています．

割り込みベクタのダンプ・リストを**リスト 1-1** に示します．

◈ 割り込みの種類

8086 の割り込みには，CPU 自身が発生する割り込みと，ハードウェアによる割り込みと，ソフトウェアによる割り込みがあります．

CPU 自身が発生する割り込みには，除算エラー，シングル・ステップ，INTO 命令等があり，CPU により割り込みの使用方法が決まっています．ハードウェア割り込みは，CPU 外部から割り込み要求(例えばタイマ LSI)があった場合に処理される割り込みです．ソフトウェア割り込みは，ユーザがプログラムの中から任意に発生させられる割り込みで，サブルーチン的な感覚で使用できます．

◈ 内部処理割り込み（CPU 自身が発生する割り込み）

CPU により使用方法が決められている割り込みには以下の 4 種類があります．

① 割り込み番号 0：除算のときに 0 で割り算を行った場合に発生

② 割り込み番号 1：シングル・ステップ動作をしているときに発生

③ 割り込み番号 3：ブレーク・ポイントのセットに使用

④ 割り込み番号 4：INTO 命令のときに発生

シングル・ステップ割り込みは，CPU が 1 命令実行するたびに割り込みがかかります．コントロール・フラグの TF：トラップ・フラグ(D8)をセットすることで，シングル・ステップ動作を開始します．プログラムのデバッグ等に使用するものです．

INTO 命令割り込みは，オーバフロー・フラグがセットされているときに，INTO 命令を実行させるとかかる割り込みです．オーバフロー・フラグがセットされていなければ，INTO 命令が実行されても割り込みはかかりません．

◈ ハードウェア割り込み

CPU 外部からハードウェアで要求できる割り込みには，NMI 割り込みと INT 割り込みがあります．NMI 割り込みはプログラムからマスクできない割り込みで，PC98 シリーズではメモリのパリティ・チェックに使用されています．

割り込み番号 2：NMI 割り込みに使用

INT 割り込みは，プログラムからマスクできる割り込みで，PC98 シリーズでは，割り込みコントローラ(μPD8259A)を 2 個使用して，全部で割り込み数を 14 個に拡張して使用しています．PC98 シリーズで割り込み処理の中心になる部分です．

割り込み番号　8～14：μPD8259A(マスタ)により割り込み

割り込み番号 16～21：μPD8259A(スレーブ)により割り込み

◈ ソフトウェア割り込み

ソフトウェア割り込みは，プログラムの中から任意にかけられる割り込みで，

INT　××　(××は割り込み番号)

という命令で，割り込み実行されます．感覚的にはサブルーチンと同様ですが，セグメントを越えて簡単に呼び出せるために，BIOS の呼び出し等に多用されています．

使用に際しては，割り込み番号 00H～1FH まではシステムが，ハードウェア割り込みや BIOS 呼び出しに使用しています．20H～3FH まではシステム予約とされ，ユーザが自由に使用できる割り込みは 40H～7FH までとされています．

ソフトウェア割り込みの使用状況は，立ち上げるシステムによって変わります．N_{88}BASIC 等では，BAISC の処理等のために，80H～F0H までを使用しています．MS-DOS では，20H～3FH までを MS-DOS を使用し，さらに，常駐する FEP 等がその他の割り込みを勝手に使用します．

◈ 拡張スロットの割り込み制御信号

拡張スロットで使用できる割り込み制御信号には，拡張メモリ用パリティ・チェック用の NMI 割り込みと，INT 割り込みを μPD8259A で拡張した割り込みのうち 8 種類を使用できます．そのうち，通常は，FDD，HDD，マウス等で 4 種類使いますが，後はユーザが自由に使えるものも多くあります．また，オプション・ボードでも，**図 1-18** のようにこれらの割り込みを使用します．

INT_3は一般的にはハード・ディスク・ドライブ・インターフェースで使用されます．PC9801-27(SASI)も，PC9801-55(SCSI)も標準では，この割り込みを使用します．内蔵の IDE 用インターフェースでも INT_3を使用しているようです．

割り込みを INT_3以外に変更できるものもあります．例えば，SASI と SCSI の二つの HDD ドライブ・インターフェースをつなぐ場合，片方の割り込みを INT_3以外で使用すると両方使えます．

INT_{41}/INT_{42}は，フロッピ・ディスク・ドライブで使用します．INT_{41}は 640KB 用，INT_{42}は 1MB 用です．内蔵の 640KB/1MB 両用インターフェースで，

〈図 1-18〉オプション・ボード割り込みレベル使用状況

オプション・ボード ＼ 割り込みレベル		INT_0	INT_1	INT_2	INT_3	INT_{41}	INT_{42}	INT_5	INT_6
PC9801-05	ODA I/F			◎					
PC9801-08/09	640KB FD I/F					◎			
PC9801-03/13	CMT I/F		◎						
PC9801-14	ミュージック I/F	○				○	○	○	◎
PC9801-15	1MB FD I/F						◎		
PC9801-16	68000 ボード								◎
PC9801-26/K	サウンド I/F	○				○	○	◎	○
PC9801-07/27	HD I/F				◎				
PC9801-06/19/29/K/N	GPIB I/F	○				○	○	◎	○
PC9801-36	CGMT I/F	○		◎				○	
PC9801-37	ファクシミリ・ボード	◎	○	○				○	
PC9801-50/55/U/L/92	SCSI I/F ボード	○	○	○	◎			○	○
PC9801-59/81	高速回線アダプタ	○				◎	◎	○	○
PC9861/K RS-232-C 拡張 I/F	CH2	◎	○	○	○				
	CH3	○				○	○	◎	○
PC9864/U	ネットワーク I/F	◎				○	○	○	○
PC9862/9866	通信制御アダプタ	◎				○	○	○	○
PC9871/K	マウス I/F	○	○	○	○	○	○	○	◎
本体内蔵マウス I/F		○	○	○	○	○	○	○	◎
PC9873	タッチ・スクリーン	◎	○					○	○
PC9801U-03/UV2 内蔵	サウンド I/F							◎	○
PC98XL-02	ImPP ボード	○	◎					○	
PC98XL^2-04	B4680 I/F ボード	○				○	○	◎	○
(PS98-144-XXX)	PC-UX ボード	○				○	○	○	◎
PC9801-77/78	B4680 I/F(NIB)	◎	○	○				○	○
PC9801-83/84	B4680 I/F(NIB)	○	◎	○	○	○	○	○	○
PC9801-88	R8100 インターフェース・ボード	○	◎	○	○	○	○	○	○
PC9866L	通信制御アダプタ	◎	○			○	○	○	
PC9867/9868	B4680 I/F(IOP)	○				○	○	◎	○
PC9801-82	GPIB ボード	○	○	○				○	○

◎：工場出荷時設定　○：変更可能レベル

640KB/1MB 自動切り替えモードで使用する場合はINT_{42}を使用しますので，普通両用ドライブを搭載している機種では，INT_{42}は拡張スロットには出ていません．

INT_5は，サウンド・ボード(FM 音源ボード)で使用されます．サウンド・ボード内蔵の機種ではINT_5に固定されているものもあります．

INT_6は一般的にはマウス・インターフェースで使用されます．INT_6以外に変更できる機種もありますが，98NOTE や最近の機種では固定されているものもあります．

トランジスタ技術 SPECIAL No.29
好評発売中
B5判 160頁
定価1,540円(税込)
特集 マイコン独習Z80完全マニュアル
手作りの原点から実用ソフトの作成まで
CQ出版社 〒170 東京都豊島区巣鴨1-14-2 販売部☎(03)5395-2141 振替 00100-7-10665

システム共通領域

0000：0400H～05FFH にシステム共通領域というエリアがあり，システムの情報，設定の識別や BIOS 等のワーク・エリアになっています．この領域を調べることで，機種依存性のある機能の有無を調べ，使用するか否かを決定できます．

この領域は参照するだけで，書き込みを行ってはいけません．

以下にシステム共通領域の機能を示します．

0400H

D_0	
D_1	
D_2	
D_3	CPU　0＝V30/V50　1＝V33
D_4	
D_5	1＝レジューム有効(E)
D_6	
D_7	RAM ドライブ 1＝搭載

0401H　拡張 RAM サイズ(128K 単位)(E)

0404H

～　リアル・モード復帰時の SP

0405H

0406H

～　リアル・モード復帰時の SS

0407H

0458H

D_0	CPU　0＝386DX　1＝386SSX
D_1	
D_2	
D_3	CPU　1＝486SX(PC-H98 のみ)
D_4	
D_5	
D_6	0＝その他　1＝マルチ・メディア(PC98GS)
D_7	0＝9801-BUS　1＝NESA-BUS

045AH

D_0	0	486SX-16	0	386SX-10
D_1	0		0	
D_2	0		0	
D_3	0		0	
D_4	0		0	
D_5	1		0	
D_6	0		1	
D_7	0		0	

045BH

D_0	
D_1	
D_2	タイム・スタンパ 1＝搭載
D_3	
D_4	
D_5	
D_6	
D_7	OUT　5FH,AL　ウェイト　1＝搭載

045CH

D_0	
D_1	
D_2	
D_3	
D_4	
D_5	CRT 垂直同期周波数＊1
D_6	1＝256 色表示可能機種
D_7	1＝486SX(PC-H98 以外)

＊1　D_5＝ノーマル・モード時 0＝24.83kHz，1＝31.47kHz
　　　ハイレゾ・モード時 0＝32.84kHz，1＝50.0kHz

0480H　CPU 種別フラグ

D_0	CPU	0	V30/286 (XA)	1	286	1	386/486/Pentium
D_1		0		0		0	
D_2							
D_3							
D_4	FD モータ制御　1＝可能(1MB/1.44MB)						
D_5							
D_6							
D_7							

0481H　システム環境フラグ

D_0	
D_1	
D_2	
D_3	キーボード＊1
D_4	
D_5	ソフトウェア EMS　1＝使用可能
D_6	キーボード＊1
D_7	

＊1

D_3	D_6	
0	0	旧式
0	1	RA,RS,RX,ES,EX,DA,DS,DX,CS,FA,FS,FX,US,BX,BA,Ap,As,Ae,Ce,Af,RL,GS
1	0	ノート/ラップトップ DIP SW_{2-7} ON
1	1	ノート/ラップトップ DIP SW_{2-7} OFF

0482H　SCSI　HDD 接続フラグ(E)

0484H　HDD 環境フラグ

D_0	CPU 1＝386SX/SL				
D_1	CPU	0	386-10	0	286-12 386-20
D_2		0	386-10/12/16	0	
D_3		0		0	

D_4	内蔵 SASI　HDD 1＝あり　(注 1)				
D_5	内蔵 SCSI　HDD 1＝あり　(注 1)				
D_6	DMA チャネル	0	CH 0	1	CH 1　(注 1)
D_7		0		0	

(注 1)EPSON PC386GS/GE 以降で DMA 変更可能な機種

0486H

〜　CPUのバージョン(EPSON PC486GR/GF以降)

0487H

0488H　RAM ドライブ接続フラグ

D_0	1MB	#0	1＝接続あり
D_1		#1	
D_2		#2	
D_3		#3	
D_4	640KB	#0	
D_5		#1	
D_6		#2	
D_7		#3	

0492H　1MB/640KB モード・チェンジ

D_0	1MB	#0	モード変更されたとき「1」になる
D_1		#1	キャリブレートを行うと「0」になる
D_2		#2	
D_3		#3	
D_4	640KB	#0	
D_5		#1	
D_6		#2	
D_7		#3	

0493H　1MB インターフェースのアクセス・モード

D_0	サーフェイス	#0	0＝片面，1＝両面
D_1		#1	
D_2		#2	
D_3		#3	
D_4	トラック	#0	0＝40 トラック，1＝80 トラック
D_5		#1	
D_6		#2	
D_7		#3	

0494H　1MB ドライブ接続(E)

D_0			
D_1			
D_2			
D_3			
D_4	1MB	#0	1＝接続あり
D_5		#1	
D_6		#2	
D_7		#3	

04ACH

〜　拡張ROMイニシャルのポインタ・オフセット・アドレス

04ADH

04AEH

〜　拡張ROMイニシャルのポインタ・セグメント・アドレス

04AFH

04B0H

〜　拡張デバイスのセグメント値の上位8ビット(16個分)

04BFH

04D0H

〜　拡張 ROM　ID(16 個分)

04DFH

0500H　システム・フラグ 1

D_0	システム既定ビット 1
D_1	0＝640KB I/F　1＝1MB I/F
D_2	640KB AI 検出　1＝許可
D_3	VSYNC(?)
D_4	拡張 RAM　1＝あり(?)
D_5	キー・バッファ・オーバフロー時のブザー　0＝鳴らす
D_6	システム既定ビット 2(DISK BASIC)
D_7	ブート種別　0＝コールド・ブート　1＝ウォーム・ブート

0501H

D_0	メモリ・サイズ	0	128 KB	1	256 KB	0	384 KB	1	512 KB	0	640 KB
D_1		0		0		1		1		0	
D_2		0		0		0		0		1	
D_3	0＝ノーマル・モード　1＝ハイレゾ・モード										
D_4	0	PC9801,XA	0	PC9801,XA, U2,LT 以外	1	U2,LT					
D_5	0		1		1						
D_6	CPU 0＝8086/286　1＝V30										
D_7	システム・クロック 0＝5/10MHz 1＝8MHz										

0502H

〜　キーボード・バッファ(16 文字分)

0521H

0522H

〜　キーボード変換テーブル・オフセット・アドレス

0523H

0524H

〜　キーボード・バッファ・データ読み込みポインタ

0525H

0526H

〜　キーボード・バッファ書き込みポインタ

0527H

0528H　キーボード・バッファ・データ・ワード数

0529H　KB ERROR(?)

052AH

〜

0539H　キーボード・キー押下状態ビット・マップ(16 ビット)

053AH　特殊キー押下状態

D_0	SHIFT	0＝押下　1＝開放
D_1	CAPS	
D_2	カナ	
D_3	GRPH	

D_4	CTRL
D_5	
D_6	
D_7	

053BH　1行中の表示ライン数−1

053CH　CRT ステータス・フラグ

D_0	画面モード　0=25行　1=20行
D_1	画面モード　0=80桁　1=40桁
D_2	アトリビュート　0=バーチカル・ライン　1=簡易グラフィック
D_3	漢字CGアクセス・モード　0=コード・アクセス　1=ドット・アクセス
D_4	
D_5	
D_6	
D_7	CRT タイプ　0=標準 CRT　1=高解像 CRT

053DH　CRT BIOS WORK

053EH

～　CRT CONT. DATA ADDRESS

0541H

0542H

～　CRTV VECTOR

0545H

0546H　CR FONT

0547H　GDC SCROLL AREA

0548H

～　PRINT VRAM ADDRESS

0549H

054AH

～　LINE 200/400

054BH

054CH　グラフィック・ステータス

D_0	0=標準モード　1=拡張モード
D_1	1=グラフィック・チャージャあり
D_2	1=16色ボードあり
D_3	1=ユーザ定義文字188文字
D_4	
D_5	
D_6	0=標準 CRT　1=高解像 CRT
D_7	0=グラフィック画面表示中　1=グラフィック画面表示停止

054DH　ドット・オペレーション・モード

D_0	描画モード	0	Replace	1	Complement	0	Clear	1	Set
D_1		0		0		1		1	
D_2	GDC クロック　0=2.5MHz　1=5MHz								
D_3	0=フラッシュレス　1=フラッシュ								
D_4									
D_5	GDC5MHz　0=禁止　1=許可								
D_6	1=拡張グラフィック描画機能あり								
D_7									

054EH

～　GDC LINE STYLE / FONT

0555H

0556H

～　RS-232-C 受信バッファ・オフセット・アドレス

0557H

0558H

～　RS-232-C 受信バッファ・セグメント・アドレス

0559H

055AH　ODA PRINTER SHIFT

055BH　データ・シフト・ステータス

D_0				
D_1				
D_2	CH	#2	0=SI 1=SO	CH #0 内蔵 RS-232-C
D_3		#1		CH #1 増設 RS-232-C 2
D_4		#0		CH #2 増設 RS-232-C 3
D_5	CH	#2	0=SI/SO 無効　1=SI/SO 有効	
D_6		#1		
D_7		#0		

055CH　1MB インターフェース FDD 接続

D_0	#0	1MB・I/F の接続　1=接続あり
D_1	#1	
D_2	#2	
D_3	#3	
D_4	#0	(2D-FDD)
D_5	#1	
D_6	#2	
D_7	#3	

055DH　HDD／FDD インターフェース接続

D_0	#0	HDD 接続　1=接続あり
D_1	#1	
D_2		
D_3		
D_4	#0	640KB・I/F の接続　1=接続あり
D_5	#1	
D_6	#2	
D_7	#3	

055EH　1MB インターフェース FDD インタラプト

D_0	#0	1MB・I/F　1=インタラプトあり
D_1	#1	
D_2	#2	
D_3	#3	
D_4		
D_5		
D_6		
D_7		

055FH　HDD／FDD インタラプト

D_0	#0	HDD インタラプト　1=あり
D_1	#1	

D_2		
D_3		
D_4	#0	640KB・I/F インタラプト　1=あり
D_5	#1	
D_6	#2	
D_7	#3	

0560H　2D-FDD 0=NON FF=1SIDE EF=2SIDE

0561H　2D-FDD OPRATION MODE

0562H
〜　2D-FDD TIMEOUT COUNTER
0563H

0564H
〜　1MB・I/F, μPD765 リザルト・ステータス
0583H　(4ドライブ分)

0584H　システムがブートしたデバイス

D_0	UA_0	DA	0FH=2HD FDD(640KB I/F)　UA　0=#1
D_1	UA_1		09H=2HD FDD(1MB I/F)　1=#2
D_2	UA_2		08H=HDD(SASI)　2=#3
D_3	UA_3		07H=2DD FDD(640KB I/F)　3=#4
D_4	DA_0		01H=2DD FDD(1MB I/F)
D_5	DA_1		0AH=HDD(SCSI)
D_6	DA_2		
D_7	DA_3		

0585H　HDD STATUS

0586H
〜　ハード・ディスク・コントローラからのエラー情報
0589H

058AH
〜　インターバル・タイマ BIOS 用　タイマ値
058BH

058EH
〜　GRAPHICS BIOS/LIO WORK
058FH

05C0H　ディップ・スイッチ SW_2(?)

D_0	SW_{2-1}	0=ON 1=OFF
D_1	SW_{2-2}	
D_2	SW_{2-3}	
D_3	SW_{2-4}	
D_4	SW_{2-5}	
D_5		
D_6		
D_7		

05C1H　RS-232-C　DEL コマンド

D_0	CH #0	0=何もしない 1=メモリ・スイッチ SW_{3-7}の設定による
D_1	CH #1	
D_2	CH #2	
D_3		

05C2H
〜　GRAPHICS BIOS POINTER
05C5H

05C6H
〜　キー・コード変換テーブル・ベース状態オフセット・
05C7H　アドレス(EPSON-PC486GR/GF 以降)

05C8H
〜　キー・コード変換テーブル・ベース状態セグメント・
05C9H　アドレス(EPSON-PC486GR/GF 以降)

05CAH　640KB インターフェースのアクセス・モード

D_0	サーフェイス	#0	0=片面　1=両面
D_1		#1	
D_2		#2	
D_3		#3	
D_4	トラック	#0	0 = 40 トラック　1 = 80 トラック
D_5		#1	
D_6		#2	
D_7		#3	

05CBH　640KB インターフェース FDD モータ OFF タイマ・カウンタ

05CCH
〜　640KB・FDD コマンド・パラメータ・ポインタ・
05CDH　オフセット

05CEH
〜　640KB・FDD コマンド・パラメータ・ポインタ・
05CFH　セグメント

05D0H　640KB インターフェース FDC リザルト・ステータス ST_0

05D1H　640KB インターフェース FDC リザルト・ステータス ST_1

05D2H　640KB インターフェース FDC リザルト・ステータス ST_2

05D3H　640KB インターフェース FDC リザルト・ステータス C

05D4H　640KB インターフェース FDC リザルト・ステータス H

05D5H　640KB インターフェース FDC リザルト・ステータス R

05D6H　640KB インターフェース FDC リザルト・ステータス N

05D7H　5 インチ FDD モータ OFF タイム・カウンタ

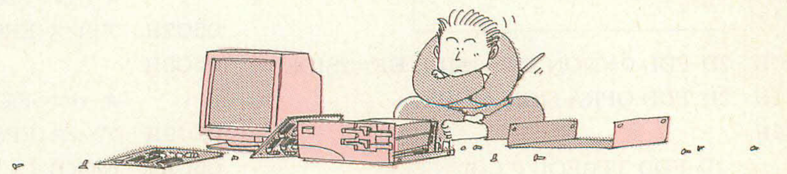
05D8H　シーク・リキャリブレート ST_0# 0〜3
〜　シーク・リキャリブレート PCN_0# 0〜3
05DFH

05F8H
〜　1MB・FDD ディスク・パラメータ・ポインタ・
05F9H　オフセット

05FAH
〜　1MB・FDD ディスク・パラメータ・ポインタ・
05FBH　セグメント

05FEH
〜　BASIC　LIO　DATA セグメント値
05FFH

2 ハードウェアの詳細



割り込みコントローラ

▶使用 LSI
μPD8259A 相当
▶ I/O アドレス
00H，02H(マスタ)
08H，0AH(スレーブ)
▶初期設定命令
ICW_1=11H(マスタ/スレーブ)
ICW_2=08H(マスタ)，ICW_2=10H(スレーブ)
ICW_3=80H(マスタ)，ICW_3=07H(スレーブ)
ICW_4=1DH(マスタ)，ICW_4=09H(スレーブ)

◈ 割り込みコントローラのハードウェア

PC98 シリーズでは，ハードウェア割り込み制御に2個の μPD8259A 相当品(以下 8259 と略す)を使用しています．8259 は1個で8本の割り込みを処理でき，さらに直列に接続(マスタとスレーブ)することで，計14(15)本の割り込みが処理できるようになります(図2-1)．

マスタ側の 8259 が，スレーブ用の割り込み受け付けのために1チャネル取られて7個，スレーブ側の IRQ_7は未使用になっているので7個あります．接続は図2-2のようになっています．

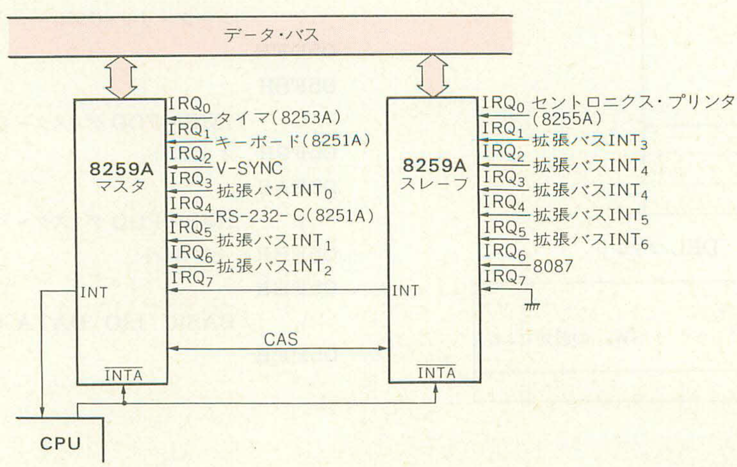
割り込みの使用状況は，本体内部で6本，拡張バスに8本です．PC9821Ap/As/Ae/Af/Ne では IR_{15}(スレーブの IRQ_7)をシステム・タイマに使用しています．

本体内部には，タイマ，キーボード，CRTV，RS-232-C，プリンタ，NDP 等に使用されています．拡張バス用に割り当てられた中から，フロッピ・ディスク・インターフェース(2DD/2HD)や，ハード・ディスク・インターフェース，マウス・インターフェースで計4本使用します．残りの割り込みは，ユーザが独自の拡張ボードで使用できます．

また，フロッピ・ディスク・インターフェースの割り込みレベルは固定になっていますが，ハード・ディスク・インターフェースと，マウス・インターフェースは，機種によっては割り込みレベルを変更することができます(図2-3)．

CPU と 8259 との割り込み制御線は，割り込み要求(INTR)と，割り込み許可(INTA)の2本だけです．これで，14(15)個もの割り込みを制御できるのは，8259 が割り込みを受け付けると，割り込み要求番号に対応した，割り込みベクタ番号を CPU に対して送

〈図2-2〉
2個の割り込みコントローラの接続図



〈図 2-1〉割り込みコントローラの I/O アドレス一覧

デバイス名	命　　令	READ/WRITE	I/O ポート・アドレス	データ D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	備　考
マスタ	ICW_1	W	00	0	0	0	1	LTIM	0	S	1	S=0
	ICW_2	W	02	T_7	T_6	T_5	T_4	T_3	0	0	0	T_7~T_3 =00001
	ICW_3	W	02	1	0	0	0	0	0	0	0	
	ICW_4	W	02	0	0	0	SFNM	BUF	1	0	1	BUF =1
	OCW_1	W	02	M_7	M_6	M_5	M_4	M_3	M_2	M_1	M_0	
	OCW_2	W	00	R	SL	EOL	0	0	L_2	L_1	L_0	
	OCW_3	W	00	0	ESMM	SMM	0	1	P	RR	RIS	
	ポール・モード	R	00	I	X	X	X	X	W_2	W_1	W_0	
	IRR リード	R	00	IR_7	IR_6	IR_5	IR_4	IR_3	IR_2	IR_1	IR_0	
	ISR リード	R	00	IS_7	IS_6	IS_5	IS_4	IS_3	IS_2	IS_1	IS_0	
	IMR リード	R	02	M_7	M_6	M_5	M_4	M_3	M_2	M_1	M_0	
スレーブ	ICW_1	W	08	0	0	0	1	LTIM	0	S	1	S=0
	ICW_2	W	0A	T_7	T_6	T_5	T_4	T_3	0	0	0	T_7~T_3 =00010
	ICW_3	W	0A	0	0	0	0	0	1	1	1	
	ICW_4	W	0A	0	0	0	SFNM	BUF	0	0	1	BUF =1
	OCW_1	W	0A	M_{15}	M_{14}	M_{13}	M_{12}	M_{11}	M_{10}	M_9	M_8	
	OCW_2	W	08	R	SL	EOI	0	0	L_2	L_1	L_0	
	OCW_3	W	08	0	ESMM	SMM	0	1	P	RR	RIS	
	ポール・モード	R	08	I	X	X	X	X	W_2	W_1	W_6	
	IRR リード	R	08	IR_{15}	IR_{14}	IR_{13}	IR_{12}	IR_{11}	IR_{10}	IR_9	IR_8	
	ISR リード	R	08	IS_{15}	IS_{14}	IS_{13}	IS_{12}	IS_{11}	IS_{10}	IS_9	IS_8	
	IMR リード	R	0A	M_{15}	M_{14}	M_{13}	M_{12}	M_{11}	M_{10}	M_9	M_8	

出することで，CPU がそれを読み取り，対応する割り込みベクタの内容のアドレスを呼び出すことで実現しています．

8259 が送出する割り込みベクタ番号は，8259 を初期化するときに設定します．つまり，割り込み処理用に割り当てられているベクタ番号 08H～17H は，再初期設定しなおせば可変可能です．しかし，変更する意味はありません．

◈ 割り込みコントローラの制御

8259 は，「IMR, IRR, ISR」の三つのレジスタと，初期設定用の四つのイニシャライズ・ワード，コマンド設定用の三つのコマンド・ワードがあり，それらを二つの I/O アドレスからアクセスします．マスタ用 8259 が 00H, 02H 番地で，スレーブ用が 08H, 0AH になります．ICW_1/OCW_2/OCW_3は書き込むときのデータ(ビット 3, 4)の設定で決定されます(図 2-4)．

▶割り込みマスク・レジスタ(IMR)

アドレス：02H(マスタ)

　　　　　0AH(スレーブ)/リード・ライト

割り込みマスク・レジスタ IMR は，コマンド・ワードの割り込みマスク・ワード OCW_1を格納するところです．このレジスタのビットが“1”ならば，割り込み要求があっても 8259 は受け付けません．

割り込みのマスクには，他にも CPU の命令の「CLI, STI」があります．これは，割り込みすべてを禁止しますが，IMR では個別に割り込みのマスクを行います．

また，割り込みを発生するデバイス自身にも，割り込みのマスクができるものも多くあります．

▶割り込み要求レジスタ(IRR)

アドレス：00H(マスタ)

　　　　　08H(スレーブ)/リード・OCW_3の RIS=0

コマンド・ワード OCW_3の RIS が“0”のときに有効になります．通常は，IRR を読み出す前に「00H(マスタ)/08H(スレーブ)に 0AH」を書き込みます．

割り込み要求レジスタ IRR は，8 レベルの割り込み入力のうち，現在割り込み要求を出しているレベルを示す情報を持っています．このレジスタのビット(D_0～D_7)は，割り込み要求入力(IR_0～IR_7)に対応しています．

▶インサービス・レジスタ(ISR)

アドレス：00H(マスタ)，

　　　　　08H(スレーブ)/リード・OCW_3の RIS=1

コマンド・ワード OCW_3の RIS が“1”のときに有

〈図 2-3〉割り込みレベルとベクタ番号

デバイス名	割り込み要求信号	レベル	割り込み名	ベクトル・データ T_7	T_6	T_5	T_4	T_3	T_2	T_1	T_0	ベクタ番号
μPD8259A /μPD71059 (マスタ)	IR_0	0	タイマ	0	0	0	0	1	0	0	0	08
	IR_1	1	キーボード	0	0	0	0	1	0	0	1	09
	IR_2	2	CRTV	0	0	0	0	1	0	1	0	0A
	IR_3	3	拡張バス INT_0	0	0	0	0	1	0	1	1	0B
	IR_4	4	RS-232-C	0	0	0	0	1	1	0	0	0C
	IR_5	5	拡張バス INT_1	0	0	0	0	1	1	0	1	0D
	IR_6	6	拡張バス INT_2	0	0	0	0	1	1	1	0	0E
	IR_7	7	スレーブ	0	0	0	0	1	1	1	1	0F
μPD8259A /μPD71059 (スレーブ)	IR_8	8	プリンタ(70116)/NDP(80286/386)	0	0	0	1	0	0	0	0	10
	IR_9	9	拡張バス INT_3(固定ディスク)	0	0	0	1	0	0	0	1	11
	IR_{10}	10	拡張バス INT_{41}(640KB FD)	0	0	0	1	0	0	1	0	12
	IR_{11}	11	拡張バス INT_{42}(1MB FD)	0	0	0	1	0	0	1	1	13
	IR_{12}	12	拡張バス INT_5	0	0	0	1	0	1	0	0	14
	IR_{13}	13	拡張バス INT_6	0	0	0	1	0	1	0	1	15
	IR_{14}	14	NDP(70116)/なし(80286/386)	0	0	0	1	0	1	1	0	16
	IR_{15}	15	(システム・タイマ)	0	0	0	1	0	1	1	1	17

* マスタおよびスレーブの T_7～T_3は，それぞれ独立にプログラムで値をセットする
* IR_8および IR_{14}は CPU によって変化する．また，80287/387 の割り込みは 80286/386 に直結されているため，80287/387 の割り込みは 8259 では制御できない
* マウスの割り込みレベルなどは，ハイレゾ・モード時は INT_2(IR_6)に固定されているが，ノーマル・モード時ではストラップ・スイッチなどにより INT_0～INT_6のいずれかを選択可能(規定値は INT_6)．98NOTE，PC9801BA，BX，PC9821Ap，As，Ae，Af では INT_6(IR_{13})に固定されている
* IR_{15}は不用意にマスクしないこと
* IR_{15}は PC9821Ap，As，Ae，Af，Ne にてシステム・タイマに使用される．その他の機種では何もせずにリターンする

〈図 2-4〉I/O アドレスとレジスタ

アドレス マスタ/スレーブ	データ D_4	D_3	READ/WRITE	レジスタ/コマンド	備考
00H/08H	1	0	WRITE	ICW_1	初期設定
02H/0AH	×	×	WRITE	ICW_2，ICW_3，ICW_4	
00H/08H	0	0	WRITE	OCW_2	EOI コマンド
00H/08H	0	1	WRITE	OCW_3	IRR/ISR 選択
00H/08H	×	×	READ	IRR/ISR	OCW_3で選択
02H/0AH	×	×	WRITE	OCW_1	IMR 書き込み
02H/0AH	×	×	READ	IMR	IMR 読み込み

効になります．通常は，ISR を読み出す前に「00H(マスタ)/08H(スレーブ)に 0BH」を書き込みます．

インサービス・レジスタ ISR は，現在処理中(サービス)のすべての割り込みレベルを示す情報を持っています．IRR と同様にビットが“1”ならば，割り込みルーチンがサービス中であることを示します．

◆ 割り込みのしくみ (完全ネスト・モードの場合)

8259 には「IR_0～IR_7(割り込み要求信号入力端子)」があり，他のデバイス(タイマ LSI 等)の割り込み要求出力端子に接続されています．他のデバイスからの割り込みが発生すると，$IR_{0\sim7}$がアクティブになり，IRR レジスタの割り込みに対応するビット(0～7)が立ちます．

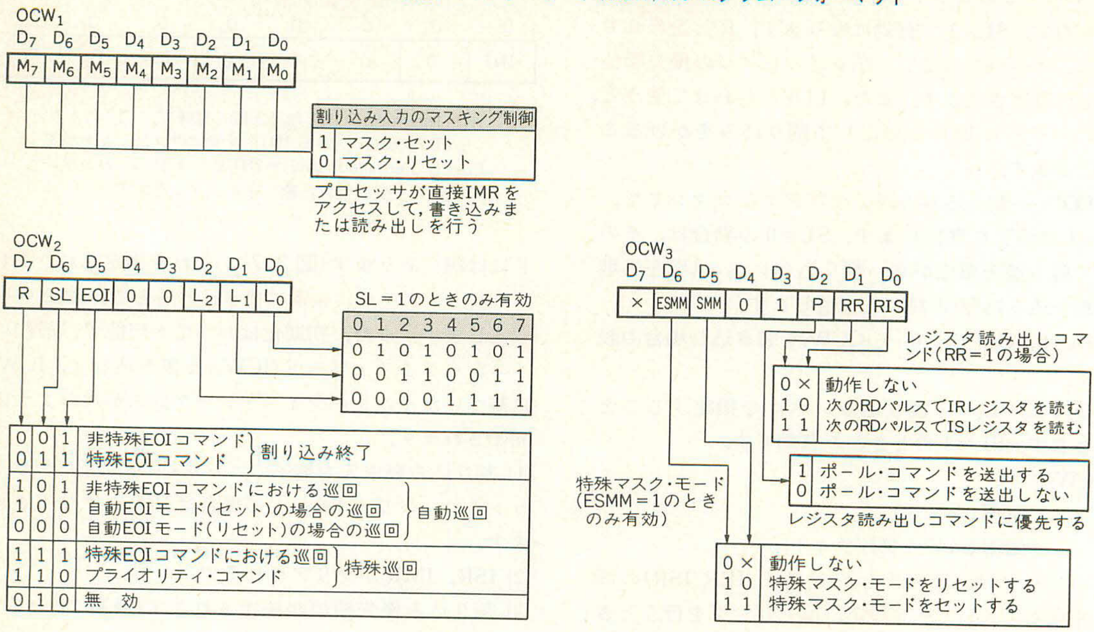
割り込みがかかると，8259 は CPU に対して割り込み要求信号(INTR)を送ると，CPU は現在実行中の処理を中止させ，割り込み許可信号(INTA)を 8259 へ返します．

8259 は，その要求されている割り込みのうち，最も優先順位の高い割り込みを ISR レジスタにセットして，あらかじめセットされている 8 ビットの 0～255 までの割り込み番号の中の一つを CPU に送ります．

CPU は，その割り込み番号に対応する割り込みベクタに書いてあるアドレスを読み込み，そのアドレスへ制御を移します．その後，IRET 命令を実行すると，割り込み発生前に処理に戻ります．

割り込みプログラム処理中に，新たに割り込みが入った場合の処理は 2 種類あります．現在の割り込みの優先順位より，新たに入った割り込みの優先順位が高ければ，いままで行われていた処理を中断して，新しい割り込み処理を行います．その反対に，現在の割り込みの優先順位より，新たに入った割り込みの優先順

〈図 2-5〉 8259A の動作コマンド・ワード(OCW)のプログラム・フォーマット



位が低ければ，現在の割り込み処理が終了するまで新しい割り込み処理は実行されません．

これらの処理は，8259が独自に管理しています．そのために，割り込み処理の終了を8259に知らせる必要があります．割り込みの終了を知らせないと，ISRレジスタの割り込みレベルに対応するビットはいつまでもクリアされず，その割り込みよりも優先度の低い割り込み要求は待たされたままになってしまいます．

このため，通常は割り込み処理の終了にEOI命令を8259に送らなければなりません．8259がEOIを受けると，そのレベルの割り込みは終了し，他の割り込みを受けられるようになります．

EOI命令はコマンド・ワードの実行で送ることができます．

◈ 特殊完全ネスト・モード

マスタにかかった割り込み処理は，マスタが直接割り込み管理できますので問題ありませんが，スレーブにかかった割り込みは，マスタから見るとすべて(7〜8個)が一つの割り込み入力に現れます．このため，マスタ側ではスレーブでの割り込みの優先順位が管理できません．

特殊完全ネスト・モードは，スレーブからきた割り込み信号はすべて処理され，その優先順位の処理はスレーブに任せることができます．この場合の割り込み終了の処理は，EOIをマスタとスレーブの両方に送らなくてはなりません．ただし，スレーブが複数の割り込みを受けている場合，ISRレジスタを監視して，スレーブのすべての割り込みが終了するのを待ってマスタにEOI命令をかける必要があります．

◈ 割り込みコントローラのコマンド実行

コマンド・ワードには三つあります(図2-5)．これを指定されたI/Oアドレス・ポートに書き込めば実行されます．IMRへの書き込み，IRR/ISRの読み出し選択等を設定します．

OCW_2/OCW_3は同じアドレスに書き込みますが，この識別は「D_3，D_4」で区別します．

▶ OCW_1

アドレス：02H(マスタ)，

　　　　0AH(スレーブ)/ライト

IMRへの書き込みをします．IRRをマスクして該当する割り込み入力からの割り込み要求を禁止します．また，完全ネスト・モードではISRもマスクします．

0＝該当割り込みレベル・マスク

1＝該当割り込みレベル許可

▶ OCW_2

アドレス：00H(マスタ)，

　　　　08H(スレーブ)/ライト

このワードは，割り込み処理の終了(EOI)を宣言するコマンドや，割り込み順位を変更するコマンドを設定します．

D_7：R……このビットは，割り込み要求の優先順位を変更(回転)するためのビットで，R＝1のときに変更します．割り込みを終了するたびに，優先順位を変更することで特定の割り込みに処理を集中させないようにできます．

D_6：SL……このビットは，L_0～L_2のビットを有効にするもので，SL=1で有効になります．Rと合わせて使うと，L_0～L_2で指定した割り込みレベルの優先順位が最低に変更されます．また，EOIと合わせて使うと，特定のレベルにEOIを指定して割り込みをかけることができます．

D_5：EOI……割り込みの終了を宣言するコマンドで，EOI=1で終了を宣言します．SL=0の場合は，その時点で最も優先順位が高い割り込むレベル(現在処理中の割り込み処理)の終了を宣言します．

D_4：D_4=0，D_3：D_3=0……OCW_2を書き込む場合の設定です．

D_0～D_2：L_0～L_2……割り込みレベルを指定することができます．SL=1のときのみ有効です．

▶ OCW_3

アドレス：00H(マスタ)，

08H(スレーブ)/ライト

このワードは，読み出しレジスタ(IRR/ISR)の設定，完全ネスト・モード設定，ポーリングを行うときに用います．

D_6：ESMM，D_5：SMM……ESMM=1のとき，SMM=1なら完全ネスト・モード設定，SMM=0なら完全ネスト・モード解除します．ESMM=0のときは何もしません．

D_4：D_4=0，D_3：D_3=1……OCW_3を書き込み場合の設定です．

D_2：P……このビットは，ポーリング・モードへの移行のときに使用します．ポーリングは割り込み処理が未対応なCPU等を使用するときに使うもので，PC98シリーズではあまり有効ではありません．CPUの割り込みを禁止して，ポーリング・データ・ポート(図2-6)を見て割り込みがかっていることを検知します．

D_1：RR/D_0：RIS……RR=1のとき，RIS=1でISR，RIS=0でIRRを読み出すレジスタとして設定します．RR=0のときは何もしません．読み出せるI/Oアドレスは，マスタは00H，スレーブは08Hです．ポーリング・モードのときはポーリング・データが読み出せます．

◈ 割り込み処理の実際

割り込み処理の具体的なプログラムを紹介しておきます．この手のプログラムはアセンブラのほうが記述しやすいのですが，最近はC言語を使用することが多く，CPUの処理速度も高速になってきてC言語でも十分な速度が得られるようになってきたので，C言語で記述してみました(リスト2-1～リスト2-3)．

◈ 割り込みコントローラの初期化

8259のイニシャライズ用の，イニシャライズ・ワードには四つあります(図2-7)．これを指定されたI/Oアドレス・ポートに順番に書き込むことで，8259を初期化します．また，初期化はいつでも可能で，最初のイニシャライズ・ワード(ICW_1)を書き込むと，ICW_1に基づいたイニシャライズ・シーケンスが次のように開始されます．

〈図2-6〉ポーリング・データ

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
INT	0	0	0	0	PL_2	PL_1	PL_0

INT(Interrupt)=INT端子と同じ意味て，"1"のとき
μPD71059が，あるINTPを受け付けたことを示す
PL_2-PL_0(Permitted Level)=INTビットが"1"のときに
有効て，受け付けた割り込みレベルを示す

(1) 割り込み要求入力端子のエッジ・トリガ回路がリセットされ，トリガ・モードの場合はIRRがクリアされます．

(2) ISR，IMRがクリアされます．

(3) 割り込み優先順位が決定されます(最高：0→7：最低)．

(4) 完全ネスト・モードは解除され，読み出しレジスタはIRRに設定されます．

(5) ICW_4レジスタがクリアされ，通常ネスト・モード，非バッファ・モード，FIコマンド・モード，CALLモードが設定されます．

ICW_1を書き込んだ後は，次に必ずICW_2以降を書き込みます．

初期設定の注意としては，PC98シリーズのハードウェアに合ったったように設定しないと正常な動作ができなくなります．

● ICW_1での注意

▶ D_0：IC_4(初期値=1)

ICW_4は，8080/8085モードと8086/8088モードを設定しなくてはならないので必ず必要です．

▶ D_1：SNGL(初期値=0)

8259はカスケード接続で二つついていますので「拡張モード」に設定します．

▶ D_2：ADI

8086/8088モードでは，この値は無効です．

▶ D_3：LTIM(初期値=0)

PC98シリーズでは，通常エッジ・トリガ・モードで使用します．エッジ・トリガ・モードは，割り込み要求信号の立ち上がりで割り込みがかかります．レベル・トリガ・モードでは，割り込み要求信号がハイ・レベルの間は何度でも割り込みがかかります．

▶ D_5～D_7：A_5～A_7

8086/8088モードでは，この値は無効です．

● ICW_2での注意

▶ D_0～D_2：A_8～A_{10}

〈リスト 2-1〉 8259(マスタ)を使用する場合の割り込み処理のプログラム

```c
/*
**
*/

#include <stdio.h>
#include <dos.h>

#define TIMER_VEC   0x08

#define TIMER_0     0x71
#define TIMER_MODE  0x77

#define PIC_IMR     0x02
#define PIC_OCW1    0x02
#define PIC_OCW2    0x00

#define EOI         0x20

    int     count   = 0;
    int     endflg  = 0;

void settimer(void);
void endtimer(void);
void interrupt timer(void);
void interrupt (*oldtimer)();

int main()
{
    unsigned int    i,j,dmy=0;

    settimer();
    printf("簡易ベンチマーク（５秒お待ち下さい）＞");
    for (i = 0; i < 60000; i++){
        for (j = 0; j < 0x1fff; j++){
            dmy += peek(j<<3,0);
            if (endflg != 0)    break;
        }
        if (endflg != 0)    break;
    }
    printf("%u¥n",i);
    endtimer();

    return 0;
}

/*
**  インターバルタイマーの設定
*/
void settimer(void)
{
    int     c10ms[2] = {24576,19968};
    int     sysclock;

    sysclock = (peekb(0,0x0501)&0x80)>>7;           /*  システムクロック    */

    outportb(PIC_OCW1,inportb(PIC_IMR)|0x01);       /*  割り込みマスク      */
    oldtimer=getvect(TIMER_VEC);                    /*  以前のベクタ保存    */
    setvect(TIMER_VEC,timer);                       /*  ベクタ設定          */

    outportb(TIMER_MODE,0x36);                      /*  ＃０　モード３      */
    outportb(TIMER_0,c10ms[sysclock]&0xff);         /*  カウンタ下位設定    */
    outportb(TIMER_0,c10ms[sysclock]>>8);           /*  カウンタ上位設定    */
    outportb(PIC_OCW1,inportb(PIC_IMR)&0xfe);       /*  割り込み許可        */
}

/*
**  インターバルタイマーの終了
*/
void endtimer(void)
{
    outportb(PIC_OCW1,inportb(PIC_IMR)|0x01);       /*  割り込みマスク      */
    setvect(TIMER_VEC,oldtimer);                    /*  ベクターを元に戻す  */
}

/*
**  インターバルタイマー割り込み部（１０ｍｓ）
*/
void interrupt timer(void)
{
    count++;
    if (count == 500)       endflg = 1;             /*  10ms x 500 = 5s     */
    outportb(PIC_OCW2,EOI);                         /*  割り込み終了        */
}
```

〈リスト 2-2〉
8259(スレーブ)を使用する場合の割り込み処理のプログラム

```
スレーブの割り込み処理

void interrupt slave_int(void)
{
    user_routine();                     /*  割り込み処理ルーチン    */

    outportb(0x08,0x20);                /*  スレーブにＥＯＩを送る  */
    outportb(0x08,0x0b);                /*  スレーブのＩＳＲを読む  */
    if (inportb(0x08) == 0){            /*  他の割り込みが無いか？  */
        outportb(0x00,0x20);            /*  マスターにＥＯＩを送る  */
    }
}
```

〈リスト 2-3〉
IRR/ISR 読み出し

```
ＩＲＲの読み方

マスター                                  スレーブ
outportb(0x00,0x0a);                      outportb(0x08,0x0a);
irr = inportb(0x00);                      irr = inportb(0x08);


ＩＳＲの読み方

マスター                                  スレーブ
outportb(0x00,0x0b);                      outportb(0x08,0x0b);
isr = inportb(0x00);                      isr = inportb(0x08);


ＩＭＲの操作

割り込み禁止
マスター                                  スレーブ
outportb(0x02,inportb(0x02)&maskbit);     outportb(0x0a,inportb(0x0a)&maskbit);
割り込み許可
マスター                                  スレーブ
outportb(0x02,inportb(0x02)&~maskbit);    outportb(0x0a,inportb(0x0a)&~maskbit);
```

8086/8088 モードでは，この値は無効です．

▶ D_3～D_7：T_3～T_7

割り込みベクタ番号の上位 5 ビットを指定します．下位 3 ビットは，レベルに合わせて自動的に 0～7 が割り当てられます．PC98 シリーズでは下記のように設定してください．

00001×××＝マスタ(08H～0FH を指定)

00010×××＝スレーブ(10H～17H を指定)

● ICW$_3$での注意

このイニシャライズ・ワードは，マスタとスレーブで設定データが異なります．マスタでは，スレーブがつながっているレベルを指定します．スレーブでは，つながっているマスタのレベル番号を指定します．PC98 シリーズでは以下のように固定されています．

マスタ　　80H　INTP$_7$のみスレーブと接続

スレーブ　03H　スレーブ番号は# 7

● ICW$_4$での注意

▶ D_0：μPM(初期値＝1)

8080/8085 モードと，8086/8088 モードを設定します．PC98 シリーズでは 8086/8088 モードに設定します．

▶ D_1：AEOI(初期値＝0)

一般的には，割り込みの終了を 8259 に知らせるためには EOI コマンドを書き込みます．しかし，このビットを 1 にすることで，自動 EOI モードとすることができます．これは CPU が割り込み受け付けした時点で自動的に EOI コマンドを出力するもので，プログラムから EOI コマンドを出力する必要がありません．反面，割り込みプログラムが処理中にでも，同じ割り込みを許可してしまうので，割り込み処理が不安定になる可能性があります．PC98 シリーズでは AEOI は使用しません．

▶ D_2：M/S

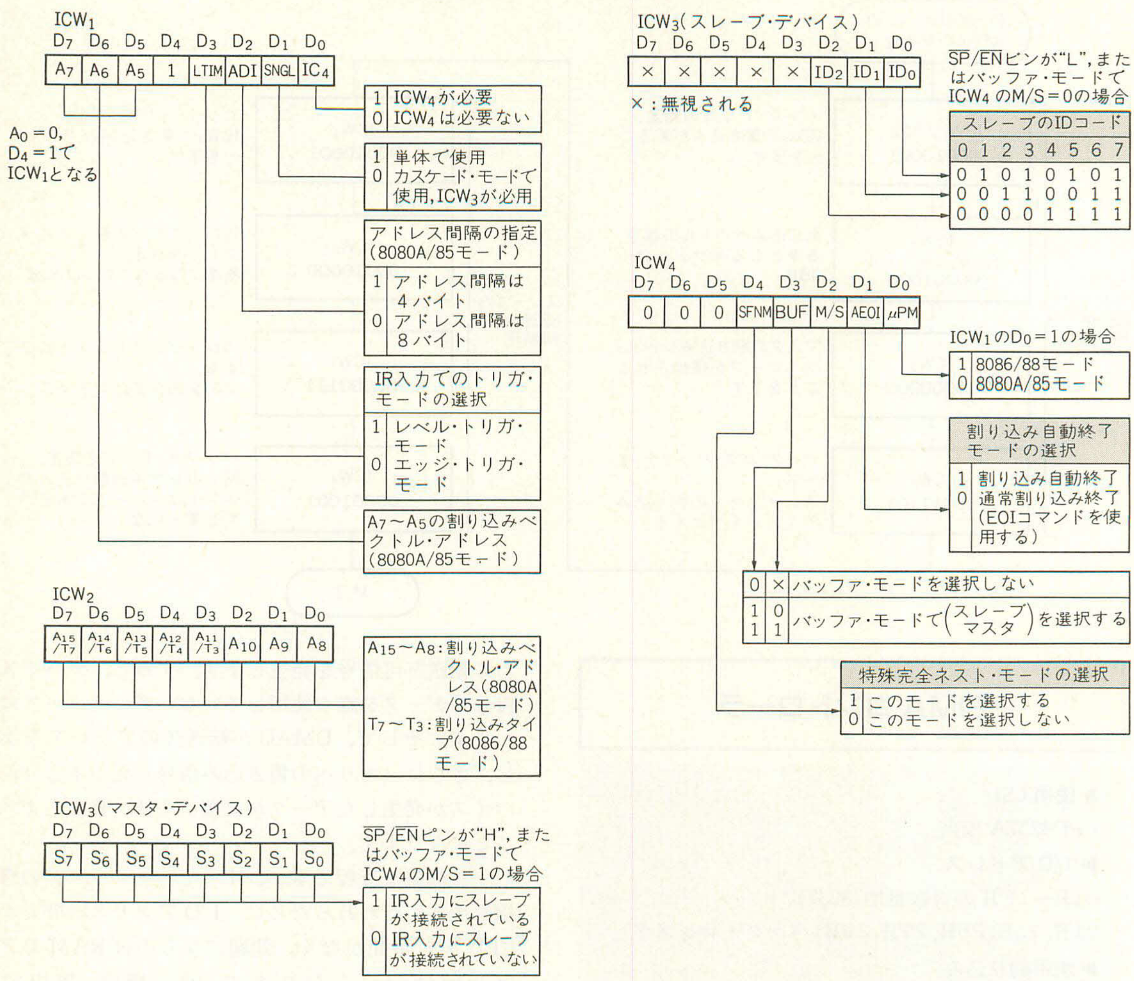
対象となる 8259 が，マスタであるかスレーブであるかを示します．“1” でマスタ，“0” でスレーブとなります．このビットが意味を持つのはバッファ・モードのときだけです．

▶ D_3：BUF(初期値＝1)

8259 と CPU との接続状態がバッファ・モードであるかどうかを指定します．

CPU に接続される素子が多くなると，データ・バスにバッファをかける必要が生じます．8259 は通常のデータ入出力動作以外に，割り込み要求が受け付けられたときに，データ・バス経由で割り込みを要求した

〈図 2-7〉 8259A の初期設定コマンド・ワード(ICW)のプログラム・フォーマット



デバイス，または処理の種類を示す割り込みベクタをCPU に送ります．このときデータ・バス・バッファの制御を 8259 の SP/EN ピンによって行います．この制御を行う場合，BUF=1 としてバッファ・モードを指定します．PC98 シリーズではバッファ・モードで使用します．

▶ D_4：SFNM(初期値：1=マスタ/0=スレーブ)

SFNM=1 で特殊完全ネスト・モードになります．

PC98 シリーズでは，マスタは特殊完全ネスト・モードで，スレーブは完全ネスト・モードに指定します．

◈ 初期化の手順

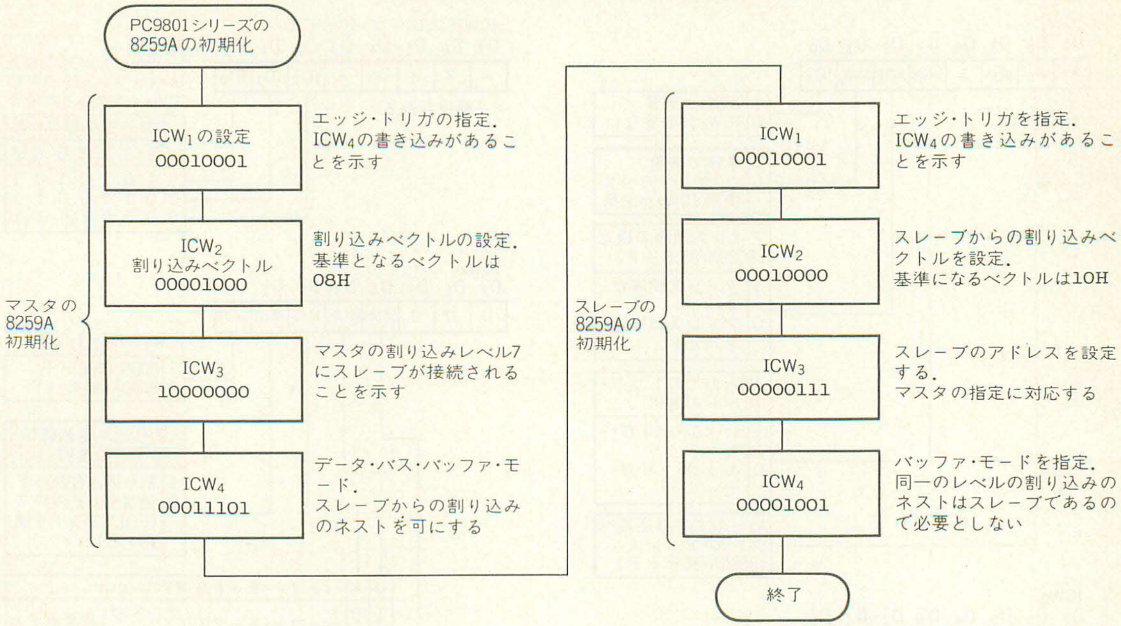
PC98 シリーズでの初期化の手順を図 2-8 に示します．8259 は周辺のハードウェアの状況に応じて，初期化コマンドが限定されますので，それに合わない設定をした場合，動作しなくなる場合がありますので注意が必要です．

初期設定の詳細は以下のようになっています．

・8086/8088 モード
・カスケード接続の「拡張モード」
・エッジ・トリガ・モード
・割り込みベクタ設定
　　マスタ　　08H～0FH を指定
　　スレーブ　10H～17H を指定
・マスタとスレーブの接続状況を設定
　　マスタ　　$INTP_7$のみスレーブと接続
　　スレーブ　スレーブ番号は#7
・自動 EOI 禁止(割り込み処理終了)
・バッファ・モード指定
・特殊完全ネスト・モード(マスタのみ)

一般的には，システム本体で用意されたデバイスを使う場合には，8259 はすでに初期化されていますから，改めて初期化したり，再設定したりする必要はほとんどありません．

〈図 2-8〉PC9801 シリーズでの初期化の手順



DMA コントローラ

▶使用 LSI

μPD8237A 相当

▶ I/O アドレス

01H～1FH の奇数番地(8237)

21H, 23H, 25H, 27H, 29H(バンク・レジスタ)

▶使用割り込み

なし

▶初期設定命令

コマンド・レジスタ＝44H(DMA 許可)

◈ DMA コントローラの役目と動作

DMA コントローラ(以下, DMAC と略す)は CPU を使用せずに, フロッピ・ディスク・インターフェース等の I/O デバイスから, メモリへ直接データを転送します. CPU を介在させて転送する場合は, デバイスがデータを持っているかどうかを確認するためにポーリングしたり, 割り込み等を使うために時間がかかります. さらに, メモリに転送する際には, 書き込み先アドレスの管理が必要です(図 2-9).

DMAC の場合は, とても高速(約 800K バイト/秒)です. I/O デバイスがデータを持つと, そのデバイスが DMAC に対して転送要求信号を発生します. DMAC は, それを見て CPU にバス・ホールド信号を出して, CPU を停止させます. CPU の停止を確認したうえで, DMAC が転送要求を出したデバイスに対して転送許可信号を発生します. すると, デバイスは自分のデータを誰も使用していないデータ・バスに流します. そして, DMAC が転送先のアドレスを出力し, さらにメモリへの書き込み信号を発生すれば, デバイスが発生したデータが直接メモリへ書き込まれることになります.

一見複雑な行程ですが, すべてハードウェアの専用線を使って行われるために, I/O アクセス時間とか, CPU 実行時間がなく, 問題になるのは RAM のアクセス時間だけとなります. 98 の場合, 実際には DMAC の動作クロック(2.5～5 MHz)が遅く, 最近の高速な CPU に負けてしまったりします.

高速化されない DMAC に見切りを付けて, 最近ではインターフェース・カード自身に専用の DMAC を持たせたもの(バス・マスタ)も登場しています.

◈ DMA コントローラのハードウェア

PC98 シリーズの DMAC には μPD8237A 相当品(以下 8237 と略す)を使用しています. これは 16 ビットのアドレス・バスと 8 ビットのデータ・バスを持ち, 単体では 64K バイトまでのメモリ・アクセスを行えます. 上位 8 ビット分のアドレス・バス(A_8～A_{15})と, データ・バスは兼用になっており, 上位 8 ビット分のアドレスを TTL 等でラッチして使用しています. また, 優先順位が付いた 4 チャネル分の DMA を持っています.

DMAC の動作中は, アドレス・バスが出力となりデータの転送を行いますが, DMAC にコマンドやパラメータを与えるときには, 下位アドレス・バスの 4 本

〈図 2-9〉 DMA コントローラの I/O アドレス一覧

	命　令	READ/ WRITE	I/O ポート・アドレス	データ							
				D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
DMAC 8237	チャネル 0 アドレス	R/W	01	A_7 A_{15}	A_6 A_{14}	A_5 A_{13}	A_4 A_{12}	A_3 A_{11}	A_2 A_{10}	A_1 A_9	A_0 A_8
	チャネル 0 カウント	R/W	03	C_7 C_{15}	C_6 C_{14}	C_5 C_{13}	C_4 C_{12}	C_3 C_{11}	C_2 C_{10}	C_1 C_9	C_0 C_8
	チャネル 1 アドレス	R/W	05	A_7 A_{15}	A_6 A_{14}	A_5 A_{13}	A_4 A_{12}	A_3 A_{11}	A_2 A_{10}	A_1 A_9	A_0 A_8
	チャネル 1 カウント	R/W	07	C_7 C_{15}	C_6 C_{14}	C_5 C_{13}	C_4 C_{12}	C_3 C_{11}	C_2 C_{10}	C_1 C_9	C_0 C_8
	チャネル 2 アドレス	R/W	09	A_7 A_{15}	A_6 A_{14}	A_5 A_{13}	A_4 A_{12}	A_3 A_{11}	A_2 A_{10}	A_1 A_9	A_0 A_8
	チャネル 2 カウント	R/W	0B	C_7 C_{15}	C_6 C_{14}	C_5 C_{13}	C_4 C_{12}	C_3 C_{11}	C_2 C_{10}	C_1 C_9	C_0 C_8
	チャネル 3 アドレス	R/W	0D	A_7 A_{15}	A_6 A_{14}	A_5 A_{13}	A_4 A_{12}	A_3 A_{11}	A_2 A_{10}	A_1 A_9	A_0 A_8
	チャネル 3 カウント	R/W	0F	C_7 C_{15}	C_6 C_{14}	C_5 C_{13}	C_4 C_{12}	C_3 C_{11}	C_2 C_{10}	C_1 C_9	C_0 C_8
	ステータス・リード	R	11	RQ_3	RQ_2	RQ_1	RQ_0	TC_3	TC_2	TC_1	TC_0
	デバイス・コントロール	W	11	AKL	RQL	EXW	ROT	CMP	DDMA	AHLD	MTM
	リクエスト・コントロール	W	13	—	—	—	—	—	SRQ	SELCH	
	マスク・コントロール	W	15	—	—	—	—	—	MSET	SELCH	
	モード・コントロール	W	17	TMODE		ADIR	SEFI	TDIR		SELCH	
	アドレス・ロー・バイト	W	19	—	—	—	—	—	—	—	—
	テンポラリ・レジスタ	R	1B	—	—	—	—	—	—	—	—
	ソフトウェア・リセット	W	1B	—	—	—	—	—	—	—	—
	クリア・オール・マスク・レジスタ	W	1D	—	—	—	—	—	—	—	—
	マスク・コントロール・レジスタ	W	1F	—	—	—	—	M_3	M_2	M_1	M_0
バンク・レジスタ	チャネル 1 バンク	W	21	A_{23}	A_{22}	A_{21}	A_{20}	A_{19}	A_{18}	A_{17}	A_{16}
	チャネル 2 バンク	W	23	A_{23}	A_{22}	A_{21}	A_{20}	A_{19}	A_{18}	A_{17}	A_{16}
	チャネル 3 バンク	W	25	A_{23}	A_{22}	A_{21}	A_{20}	A_{19}	A_{18}	A_{17}	A_{16}
	チャネル 0 バンク	W	27	A_{23}	A_{22}	A_{21}	A_{20}	A_{19}	A_{18}	A_{17}	A_{16}
	バンク・アドレス・オート・インクリメント・モード・レジスタ	W	29	0	0	0	0	M_1	M_0	CS_1	CS_0

がアドレス入力用になり，レジスタの選択をします．DMACは16個のI/Oアドレスに26個のレジスタを持ってます．

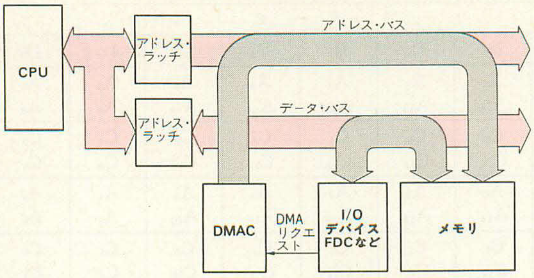
98シリーズでは，DMACのI/Oアドレスを奇数アドレスの，01H，03H，05H，…1FHの16個に割り当てていますので，DMACのレジスタをCPUがアクセスする時は，DMACのアドレス・バスとCPUのアドレス・バスが，$A_0=A_1$，$A_1=A_2$のように，一つ分ずれて接続されています．もちろん，メモリ転送する場合は，奇数アドレスも偶数アドレスも両方アクセスできなくてはなりませんから，$A_0=A_0$，$A_1=A_1$のような接続に切り替えます．

DMACを使用するI/Oデバイスは，フロッピ・ディスクやハード・ディスクのコントローラですが，それらはI/O空間の偶数アドレス(データ・バスの下位8ビット)に接続されています．DMACを使用する場合は，図2-10のように，I/OデバイスがDMACにDMAのリクエスト信号を送ります．すると，DMACはアドレスを発生させます．そのときにI/Oデバイスとメモリの間でデータ・バスを通して直接にデータ転送が行われます．

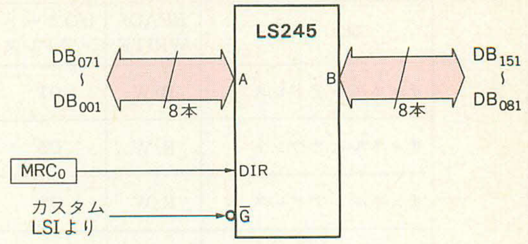
ここで，フロッピ・ディスク・コントローラ等は，データ・バスの下位8ビットに接続されていますので，そのままでは奇数アドレス(上位)のメモリとは，データ・バスが直接接続されていませんので転送ができません．

そのため，図2-11のような回路がデータ・バスに接続されていて，下位D_0〜D_7と，上位D_8〜D_{15}とを結んで，奇数アドレスと偶数アドレスとのデータ転送を可能にしています．奇数アドレスと偶数アドレスの転

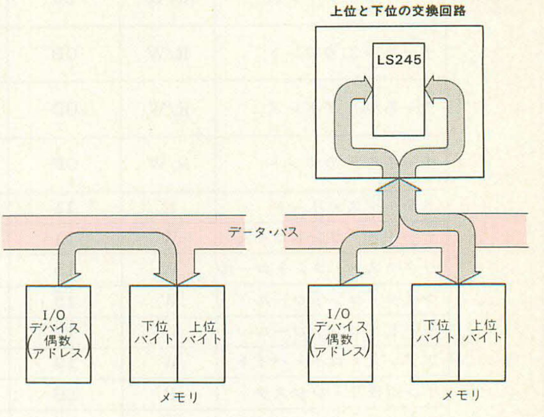
〈図2-10〉DMA 転送のしくみ



〈図2-11〉データ・バスの上位と下位を結ぶ双方向性バッファ



〈図2-12〉奇数アドレスと偶数アドレスの転送の違い



送の違いを図2-12に示します.

図2-11の回路が，奇数アドレスを持つメモリ・アクセス時に介在してくるので，DMAを使用するデバイスはすべて偶数アドレスにする必要があります．また，メモリとメモリの間の転送は，DMACのレジスタが奇数アドレスにあるため使用できません.

◈ DMA コントローラのレジスタ

DMACのレジスタは大きく分けると，アドレス・レジスタ，カウント・レジスタ，コントロール・レジスタ群に分けられます(図2-13).

● アドレス/カウント・レジスタ

▶ 01H，05H，09H，0DH：アドレス・レジスタ/ライト

アドレス・レジスタは，DMA転送用アドレスの初期値を入れます．このレジスタは一つのI/Oアドレスに2回書き込むことで，16ビット分を指定することができます.

▶ 01H，05H，09H，0DH：イフェクティブ・アドレス・レジスタ/リード・ライト

DMA転送中に有効なレジスタです.

DMA転送中のアドレスを読み書きできます．DMAが転送するたびに自動的に+1/−1されていきます．このレジスタは一つのI/Oアドレスに2回読み書きすることで，16ビット分を指定することができます.

▶ 03H，07H，0BH，0FH：カウント・レジスタ/ライト

DMA転送バイト数の初期値を入れます．転送開始時に，このレジスタの内容をイフェクティブ・カウント・レジスタへロードします．再度，CPUから書き換えるまでは値は変わりません．このレジスタは一つのI/Oアドレスに2回読み書きすることで，16ビット分を指定することができます.

▶ 03H，07H，0BH，0FH：イフェクティブ・カウント・レジスタ/リード・ライト

DMA転送中に有効なレジスタです.

DMA転送の残りバイトを読み書きできます．DMAが転送するたびに自動的に−1されていき，0からアンダ・フローした場合に，DMA転送終了となります．このレジスタは一つのI/Oアドレスに2回読み書きすることで，16ビット分を指定することができます.

▶ 11H：デバイス・コントロール・レジスタ/ライト

DMA転送における転送モード，DMA要求の許可/禁止等の制御に使用します．図2-14にフォーマットを示します.

D_0：MTM(初期値=0)……MTM=1でメモリ-メモリ間転送を行います．メモリ間転送はチャネル0→1に固定され，ソフトウェアDMA要求でスタートします．ただし，PC98シリーズではメモリ間転送はできませんので，通常はMTM=0と設定します.

D_1：AHLD(初期値=0)……MTM=1のときに，AHLD=1にすると，チャネル0のアドレスは初期値に固定されます(自動インクリメントされない)．MTM=0のときは無視されます.

D_2：DDMA……DDMA=1で，すべてのDMA要求を禁止します.

D_3：CMP(初期値=0)……CMP=1で圧縮転送を行います．圧縮転送は，DMACの上位アドレスの出力を必要なときのみ行うことで，転送のサイクル(タイミング)を縮めて高速に転送する方式です．ただし，ディマインド・モード/ブロック・モードのみ有効なので，PC98シリーズでは使用せず，通常転送(CMP=0)で使用します.

〈図2-13〉アドレス/カウント・レジスタ

チャネル	レジスタ名称	READ/WRITE	I/O アドレス
0	セット・アドレス・レジスタ	W	01H
	イフェクティブ・アドレス・レジスタ	R/W	
	セット・カウント・レジスタ	W	03H
	イフェクティブ・カウント・レジスタ	R/W	
1	セット・アドレス・レジスタ	W	05H
	イフェクティブ・アドレス・レジスタ	R/W	
	セット・カウント・レジスタ	W	07H
	イフェクティブ・カウント・レジスタ	R/W	
2	セット・アドレス・レジスタ	W	09H
	イフェクティブ・アドレス・レジスタ	R/W	
	セット・カウント・レジスタ	W	0BH
	イフェクティブ・カウント・レジスタ	R/W	
3	セット・アドレス・レジスタ	W	0DH
	イフェクティブ・アドレス・レジスタ	R/W	
	セット・カウント・レジスタ	W	0FH
	イフェクティブ・カウント・レジスタ	R/W	

D_4：ROT(初期値＝0)……ROT＝1で回転ネスト・モード，ROT＝0で固定ネスト・モードです．回転ネスト・モードは，最後にDMAサービスを受けたチャネルの優先順位が最低になるように順次変化させて，特定のチャネルが独占されるのを防ぎます．PC98は，優先順位が「0>1>2>3」で固定されている固定ネスト・モードで使用します．

D_5：EXW(初期値＝0)……EXW＝1で拡張書き込みモード，EXW＝0で遅れ書き込みモードです．EXW＝1にすると，メモリやI/O書き込みタイミングが，読み込みタイミングと同じタイミングで出力されます．場合によっては，ウェイト・サイクルを減らすことができますが，PC98シリーズでは通常使用しません．また，CMP＝1の場合は無効になります．

D_6：RQL(初期値＝1)……DRQ(DMA要求信号)アクティブ・レベルを変更します．RQL＝0でアクティブ"H"，RQL＝1でアクティブ"L"です．PC98シリーズでは，RQL＝1で使用します．

D_7：AKL(初期値＝0)……DACK(DMA許可信号)アクティブ・レベルを変更します．AKL＝0でアクティブ"H"，AKL＝1でアクティブ"L"です．PC98シリーズでは，AKL＝0で使用します．

▶ 11H：ステータス・リード・レジスタ/リード

各チャネルの，DMA要求，転送終了状態を調べます．図2-15にステータス・リード・レジスタのフォーマットを示します．

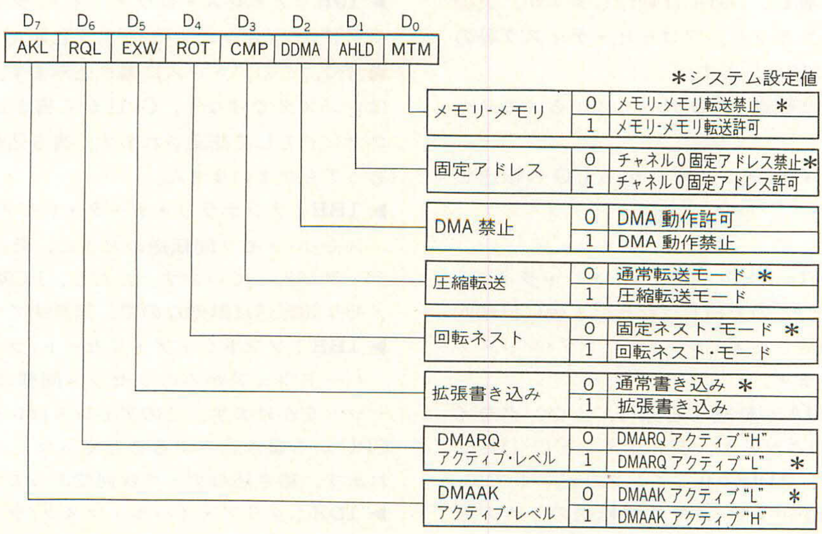
D_0〜D_3：TC_0〜TC_3……END入力，またはターミナル・カウント(転送終端)の場合は$TC_×$＝1，未終端の場合は$TC_×$＝0になります．

D_4〜D_7：RQ_0〜RQ_3……DRQ端子によって起きた，DMA要求の状態を示します．DMA要求があった場合に$RQ_×$＝1です．

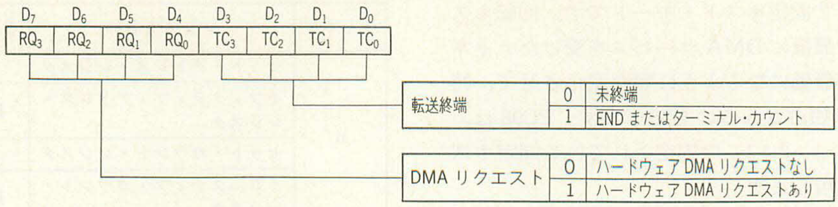
▶ 13H：リクエスト・コントロール・レジスタ/ライト(使用禁止)

ソフトウェアによるDMA要求を実行します．しかし，PC98シリーズでは，ソフトウェアによるDMA要求は禁止されています．図2-16にリクエスト・コントロール・レジスタのフォーマットを示します．

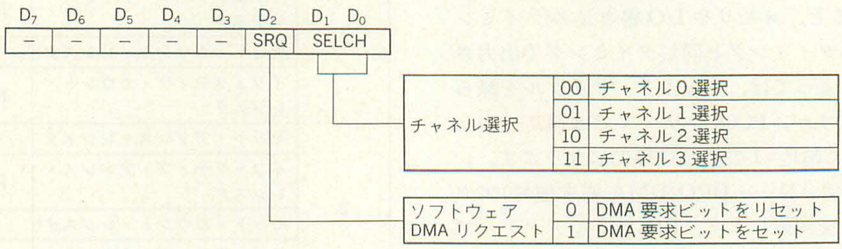
〈図2-14〉デバイス・コントロール・レジスタ



〈図 2-15〉ステータス・リード・レジスタ



〈図 2-16〉リクエスト・コントロール・レジスタ



▶ 15H：マスク・コントロール・レジスタ/ライト

図2-17に示すこのレジスタは，ハードウェア(DRQ端子)からのDMA要求の許可/禁止を設定します．設定には，チャネルごとに設定する方法と，すべてのチャネルを同時に設定する方法がありますが，「15H：マスク・コントロール・レジスタ」はチャネルごとに別々に設定します．

DMACにコマンドを与える場合は，DMACが動作中であってはいけません．そのために，コマンドを与える前にマスク・ビットをセットしておきます．

▶ 17H：モード・コントロール・レジスタ/ライト

各チャネルごとに，DMA転送動作を制御します(図2-18)．

D_0～D_1：SELCH……DMAチャネルを指定します(0～3)

D_2～D_3：TDIR……転送方向を指定します．

00＝ベリファイ転送．DMAは動作しますが，実際にメモリへ転送されません．フロッピ・ディスク等のCRCチェック等に使用します．

01＝I/O→メモリ転送．I/Oデバイスからメモリへ転送します．

10＝メモリ→I/O転送．メモリからI/Oへ転送します．

11＝使用禁止

D_4：SEFI……SEFI＝1でセルフィ・イニシャライズ・モードになります．このモードは転送終了後に自動的に，最初に初期設定したアドレス/カウンタ・レジスタの内容をロードします．

D_5：ADIR……DMAが転送する際に，カウンタをインクリメント(+1)させるか，デクリメント(−1)させるかを指定します．ADIR＝0でインクリメントです．

D_6～D_7：TMODE……I/O-メモリ間転送のときの転送モードを選びます．PC98シリーズでは「01：シングル・モード」を使用します．

00＝デマインド・モード．転送がすべて終了するか，DMA要求信号がクリアされるまで転送を続けます．その間，CPUは停止しているのでプログラムは一切動きません(割り込みも同様)．

01＝シングル・モード．1回(1バイト)転送するたびに，CPUが1マシン・サイクル動作します．DMA転送中であっても割り込み等が受け付けられます．

10＝ブロック・モード．転送がすべて終了するまで転送を続けます．その間，CPUは停止しているのでプログラムは一切動きません．メモリ間転送はブロック・モードで行います．

11＝拡張モード．DMACをカスケード接続して使う場合に使用しますが，PC98シリーズでは，DMACは一つかありませんから使用しません．

▶ 19H：アドレス・ロウ・バイト/ライト

アドレス/カウント・レジスタに新たな値を設定する場合に，このアドレスに書き込みます．このアドレスはレジスタではなく，CPUから書き込みすることでコマンドとして認識されます．書き込むデータは何であってもかまいません．

▶ 1BH：テンポラリ・データ・レジスタ/リード

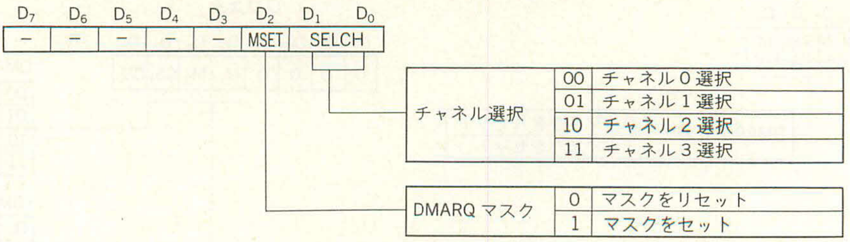
メモリ-メモリ間転送のときに，最後に転送されたデータが入っています．ただし，PC98シリーズではメモリ間転送は無効なので，無意味です．

▶ 1BH：ソフトウェア・リセット/ライト

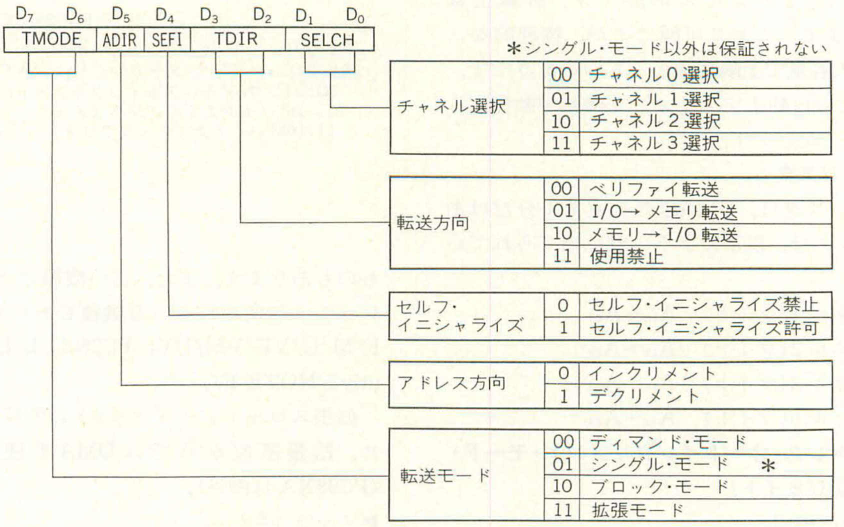
ハードウェアからのリセット同様に，DMACにリセットをかけます．このアドレスはレジスタではなく，CPUから書き込みすることでコマンドとして認識されます．書き込むデータは何であってもかまいません．

▶ 1DH：クリア・オール・マスク/ライト

〈図 2-17〉マスク・コントロール・レジスタ



〈図 2-18〉モード・コントロール・レジスタ



全チャネルのマスク・コントロール・レジスタをクリアし，DMA転送の要求を許可します．このアドレスはレジスタではなく，CPUから書き込みすることでコマンドとして認識されます．書き込むデータは何でもかまいません．

▶ 1FH：マスク・コントロール・レジスタ/ライト

ハードウェア(DRQ端子)からのDMA要求の許可/禁止を設定します．このレジスタからは，すべてのチャネルを同時に設定します(図2-19)．

◈ 64Kバイトを超えるアクセス

DMAC単体では64Kバイトまでのアクセスしかできません，それ以上のアクセスを可能にするために，A_{16}〜A_{19}までの4ビットを拡張したバンク・レジスタがあります．このレジスタは，CPUからレジスタに書き込んだデータを，DMACがメモリ・アクセスする際に，そのままアドレス・バスに出力し，1Mバイトまでのアクセスを可能にしたものです．

CPUにV30を持つ機種では，DMACとバンク・レジスタとは無関係に動作しますので，64Kバイトの境界を越えるアクセスはできません．この場合は2回に分けて，バンク・レジスタを設定しなおして転送することになります．

CPUに80286以上を持つ機種では，バンク・レジスタがA_{16}〜A_{23}までの8ビットに拡張され，16Mバイトまでのメモリがアクセス可能です．さらに，64Kバイトの境界で，自動的にバンク・レジスタがインクリメントするので，すべて，一度で転送できるようになりました．

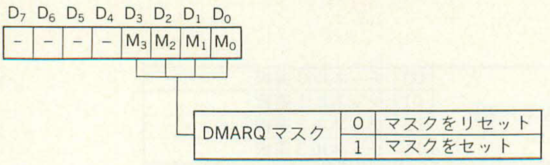
バンク・レジスタが4ビットから8ビットへ拡張されたのに従って，互換を持たせるために通常は8ビットのレジスタの上位4ビットが，"0"にマスクされています．1Mバイトを超える転送をするときには，DMAアドレス・マスク・レジスタ(0439H)を操作することで可能になります．

● DMAアドレス・マスク・レジスタ (0439H/リード・ライト)

図2-20にこれを示します．D_2以外のビットも他の機能に使われていますので，MSKビットを変更する場合は，D_0〜D_7まですべてのビットを読み取り，MSKビットのみ変更して，再び書くようにします．また，DMA転送が終了次第，MSKビットを元の状態に戻す必要があります．

PC9821Af以降の機種では，メモリ空間を16Mバ

〈図 2-19〉マスク・コントロール・レジスタ



イト以上持てるものがあります．これらの機種では，バンク・レジスタが 8+8 ビットに拡張されていて，全部で 32 ビットまでアクセス可能です．計算上は 4096M バイトまでアクセス可能ですが，物理的なメモリ増設の最大容量で上限が決まっているようです．しかし，DMA の自動インクリメント機能が動作するのは 16M バイトだけです．

● バンク・レジスタ

バンク・レジスタは，DMAC のチャネル分だけあり，I/O アドレスは，以下のように割り当てられています．

21H　チャネル 1(ライト)　A_{16}～A_{23}
23H　チャネル 2(ライト)　A_{16}～A_{23}
25H　チャネル 3(ライト)　A_{16}～A_{23}
27H　チャネル 0(ライト)　A_{16}～A_{23}

● バンク・アドレス・オートインクリメント・モード・レジスタ(29H/ライト)

図 2-21 に示します．

◈ DMA コントローラの使用状況

DMA の使用状況は，ノーマル・モード時ではどの機種でも同じような割り当てがされています．異なるのは，ハード・ディスク・ドライブ・インターフェースや，メモリ・リフレッシュに使用している機種で少し異なります．DMA の優先順位は「0」が一番高く，「3」が一番低くなり，同時に発生した場合は優先順位が高い順番に処理されます．

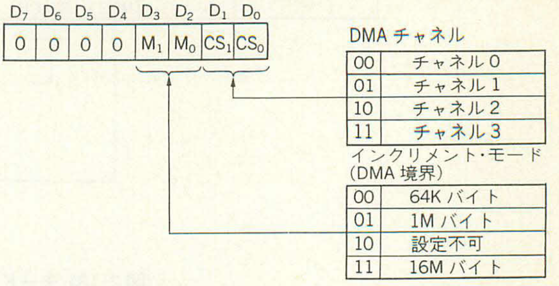
▶チャネル#0

主にハード・ディスク・ドライブ・インターフェースに使われます．ディップ・スイッチ $SW_{3\text{-}3}$で，チャネルを「0」,「1」に替えられるものもあります．

▶チャネル#1

主に未使用か使用不可のどちらかで，内蔵ハード・ディスク・ドライブ・インターフェースに使われている

〈図 2-21〉バンク・アドレス・オートインクリメント・モード・レジスタ



PC-98XA では 11 の設定は不可

インクリメント・モード：8237 から出力されるアドレスと連結してバンク・アドレスをインクリメントする
　00: バンク・アドレスをインクリメントしない
　01:1M バイトまでのインクリメント
　11:16M バイトまでのインクリメント

ものもあります．また，古い機種では，メモリ・リフレッシュに使われている機種もあります(PC9801/E/F/M/U/VF/VM/UV，PC286U/L/LE/NOTE executive/NOTE F)．

拡張スロットにチャネル#1 の信号が出ていないため，拡張基板からでは DMA を使用できません(PC98XA は例外)．

▶チャネル#2

2HD フロッピ・ディスク・インターフェースに使われています．

拡張スロットにチャネル#2 の信号が出ていないため，拡張基板からでは DMA を使用できません(PC9801/E/F/U/VF/VM/UV/CV では最も大きな番号のスロットのみ使用可能．PC98XA は例外)．

▶チャネル#3

2DD フロッピ・ディスク・インターフェースに使われています．

チャネル割り当てを図 2-22 に，使用状況を図 2-23 に示します．

◈ DMA コントローラの使用方法

DMA 使用方法のサンプル・プログラムとして，1MB フロッピ・ディスク・ドライブを読み出すプログラムを紹介します(リスト 2-4)．

〈図 2-20〉DMA アドレス・マスク・レジスタ

命　令	READ/WRITE	I/O ポート・アドレス	データ D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
DMA アドレス・マスク・レジスタ	R/W	0439	—	—	—	—	—	MSK	—	—

D_2：MSK0：A_{23}～A_{20}有効
　　　　　1：A_{23}～A_{20}無効

〈図 2-22〉 チャネル割り当て

チャネル			0	1	2	3
PC9801/E/F1,2,3/M2,3/U2/VF2/VM0,2,4/UV2/VM21			HDD	メモリ・リフレッシュ*1	1MB FD	640KB FD
PC9801T/DA/DS/DX/CS/FA/FS/FX/PC9821Ap/As/Ae/Af/PC98GS	SW_{3-3} OFF		HDD	使用不可	1MB FD	640KB FD
	SW_{3-3} ON		未使用	内蔵 HDD	1MB FD	640KB FD
PC98XA model 1,2,3/11,21,31			未使用	1MB FD	640KB FD	HDD
PC98XL model 1,2,4 PC98XL^2		ノーマル	HDD	未使用	1MB FD	640KB FD
		ハイレゾ	HDD	1MB FD	未使用	640KB FD
PC98RL*2	SW_{3-3} OFF	ノーマル	HDD	使用不可	1MB FD	640KB FD
		ハイレゾ	HDD	1MB FD	使用不可	640KB FD
	SW_{3-3} ON	ノーマル	未使用	内蔵 HDD	1MB FD	640KB FD
		ハイレゾ	未使用	1MB FD	内蔵 HDD	640KB FD
上記以外			HDD	未使用	1MB FD	640KB FD
優先順位			高 ←―――	―――――	―――――	―――→ 低

＊1：メモリ・リフレッシュは 64K バイト単位で行う

＊2：ディップ・スイッチによる DMA のチャネル切り替えは，本体内蔵固定ディスクのみ可能

〈図 2-23〉 DMA チャネル使用状況

オプション・ボード ＼ DMA チャネル		#0	#1	#2	#3
PC9801-08/09/本体内蔵	640KB FD I/F				◎
PC9801-15/本体内蔵	1MB FD I/F			◎	
システム・リザーブ			◎		
PC9801-07/27/本体内蔵	HDD I/F	◎			
PC9801-29N	GP-IB I/F	○			◎
PC9801-36	CGMT I/F	○			◎
PC9801-37	ファクシミリ・ボード	○			◎
PC98XL-02	ImPP ボード	○			◎
PC9801-55/L/U/92	SCSI I/F	◎		○	○
PC9866/L	通信制御アダプタ	◎			○
PC9801-82	GP-IB ボード	○			○

◎：工場出荷時設定　○：変更可能レベル

〈参考リスト〉 8253の動作中のカウント・データの読み取り方法

（カウンタ・ラッチ・コマンドを使用すると，その瞬間のカウンタの内容が読み取れる）

```
/*
**   ８２５３のカウントデータ読取り
*/

#define  counter0  0x71      /* 8253 couner #0 : timer */
#define  counter1  0x73      /* 8253 couner #1 : beep */
#define  counter2  0x75      /* 8253 couner #2 : rs232c */
#define  mode8253  0x77      /* 8253 mode reg. */

#define  latch0  0x00        /* counter #0 : count latch : binary */
#define  latch1  0x40        /* counter #1 : count latch : binary */
#define  latch2  0x80        /* counter #2 : count latch : binary */

unsigned int getcounter(void)
{
  unsigned int count;
  outportb(counter2, latch2); /* counter #2 */
  count=inportb(counter2);
  count=count | (inportb(counter2) << 8);
  return(count);
}
```

〈リスト2-4〉フロッピ・ディスク読み出しプログラムの例

```
/*
**  ＤＭＡ  テスト
**
**  ドライブ０，トラック０，サイド０，セクター１～８をＤＵＭＰ表示。
*/

#include <stdlib.h>
#include <stdio.h>
#include <dos.h>

#define     FDC_STAT    0x90
#define     FDC_RESULT  0x92
#define     FDC_CMD     0x92

#define     DMA_TFR_SIZE        1024*8
#define     DMA_MASK            0x15
#define     DMA_MODE            0x17
#define     DMA_STAT            0x11
#define     DMA_CLRBP           0x19
#define     DMA_CH2_BANK        0x23
#define     DMA_CH2_ADRS        0x09
#define     DMA_CH2_COUNT       0x0b

    char    *buff;

void setfdccmd(int *cmd);
void setdma(unsigned int tfrseg, unsigned int tfroff, unsigned int tfrsize);

void main(void)
{
    int     fdc_seek[] = {0x0f, 0, 0, -1};                          /*  seek    */
    int     fdc_read[] = {0x46, 0, 0, 0, 1, 3, 8, 0x35, 0xff, -1};  /*  read    */
    int     i, count;

    if ((buff = calloc(DMA_TFR_SIZE, 1)) != NULL){
        setfdccmd(fdc_seek);
        setdma(FP_SEG(buff), FP_OFF(buff), DMA_TFR_SIZE);
        setfdccmd(fdc_read);
        for (i = 0; i < DMA_TFR_SIZE; i++){
            do{
                count = inportb(DMA_CH2_COUNT);     /* DMA CH2 COUNT LOW    */
                count += inportb(DMA_CH2_COUNT)<<8; /* DMA CH2 COUNT HI     */
            }while (count >= DMA_TFR_SIZE-i);       /* DMA 転送量 ﾁｪｯｸ      */
            if (i%16 == 0)      printf("¥n%04X :", i);
            printf(" %02X", buff[i]);
        }
        printf("¥n");
        inportb(DMA_STAT);                          /*  TC bit clear    */
        free(buff);
    }
}

/*
**  ＦＤＣにコマンドをセット
**
**  コマンド（パラメータ）終了はー１
*/
void setfdccmd(int *cmd)
{
    int     stat, i=0;

    while (cmd[i] >= 0){
        do {
            stat = inportb(FDC_STAT);
        }while(((stat&0x80)==0)||((stat&0x40)!=0));
        outportb(FDC_CMD, cmd[i]);
        i++;
    }
}

/*
**  ＤＭＡにパラメータ設定
**
*/
void setdma(unsigned int tfrseg, unsigned int tfroff, unsigned int tfrsize)
{
    unsigned int    bank, adres;

    outportb(DMA_MASK, 0x06);                       /*  ｃｈ２マスク          */
    bank = tfrseg >> 12;
    adres = (tfrseg << 4) + tfroff;
    outportb(DMA_CH2_BANK, bank);                   /*  バンク      設定      */
    outportb(DMA_CH2_ADRS, adres&0xff);             /*  アドレスＬＯＷ設定    */
    outportb(DMA_CH2_ADRS, adres>>8);               /*  アドレスＨＩ  設定    */
    outportb(DMA_CH2_COUNT, tfrsize&0xff);          /*  カウントＬＯＷ設定    */
    outportb(DMA_CH2_COUNT, tfrsize>>8);            /*  カウントＨＩ  設定    */
/*  モード設定  ｼﾝｸﾞﾙ転送, ｱﾄﾞﾚｽｲﾝｸﾘﾒﾝﾄ, ｵｰﾄｲﾆｼｬﾗｲｽﾞ禁止, ﾗｲﾄ転送, CH2  */
    outportb(DMA_MODE, 0x46);
    outportb(DMA_MASK, 0x02);                       /*  ｃｈ２マスク解除      */
}
```

タイマ

▶使用 LSI
8253C 相当
▶ I/O アドレス
71H, 73H, 75H, 77H(旧機種)
71H, 75H, 77H, 3FDBH, 3FDFH
▶使用割り込み
IR_0(割り込みベクタ# 08H)

図 2-24 に I/O アドレス一覧を示します.

98 シリーズでは, タイマ用 LSI に μPD8253C の互換品(以降は 8253 と略す)を使用しています. この LSI は, 入力されたクロック(約 2〜2.5 MHz/CLK 入力)を 16 ビットのカウンタで分周して任意の周波数を作り出すことができます. また, 三つの独立した出力と三つの独立したカウンタを持ち, それぞれ六つのモードを選択することができます.

タイマ LSI 周辺回路を図 2-25 に示します.

◈ 8253 に入力されるクロック(CLK 入力)の違い

LSI に入力されるクロック(以下, CLK 入力と略す)には 2 種類あり, 拡張スロットのシステム・クロックが 8 MHz 系(CPU クロックが, 8/16/33/60/66 MHz)の時には, 1.9968 MHz, 5 MHz 系(CPU クロックが, 5/10/12/20/25/40 MHz)のときには 2.4576 MHz になります(機種によっては該当しない場合もある). タイマ LSI はこのクロックを分周して使用していますから, 8253 にどちらのクロックが入力されているか認識しないと, タイミングや周波数が変わってしまい互換性がとれなくなってしまいます. これを調べるには, プリンタ・ポート用の 8255 の 42H のビット 5 を読み取り, “1” ならば 1.9968 MHz, “0” ならば 2.4576 MHz であることがわかります.

EPSON の PC シリーズで, PC386S/G/GS/GE/P/GR/GF では, システム・クロックを変更できますので, それに合わせた CLK 入力が入ります. その他の機種では, CPU クロックが 10 MHz 以上ならば 2.4576 MHz(PC286VE の 10 MHz 時を除く), 10 MHz 未満ならば 1.9968 MHz が供給されます. NEC の 98 シリーズと若干異なりますが, プリンタ・ポート用の 8255 の 42H のビット 5 で得られる情報は同様ですから, プログラム的には互換性が保たれます.

例外として, 初代の PC9801 では, このプリンタ・ポート用の 8255 の 42H のビット 5 がプリンタ・インターフェース・コネクタの 13 ピンにつながっており,

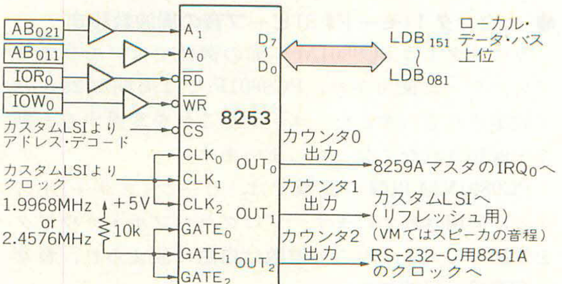
命令	READ/WRITE	I/O ポート・アドレス	データ							
			D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
カウンタ 0 へのロード	W	71	C_7 C_{15}	— —	— —	— —	— —	— —	— —	C_0 C_8
カウンタ 0 をリード	R	71	C_7 C_{15}	— —	— —	— —	— —	— —	— —	C_0 C_8
カウンタ 1 へのロード	W	73/3FDB ＊1	C_7 C_{15}	— —	— —	— —	— —	— —	— —	C_0 C_8
カウンタ 1 をリード	R	73/3FDB ＊1	C_7 C_{15}	— —	— —	— —	— —	— —	— —	C_0 C_8
カウンタ 2 へのロード	W	75	C_7 C_{15}	— —	— —	— —	— —	— —	— —	C_0 C_8
カウンタ 2 をリード 0	W	75	C_7 C_{15}	— —	— —	— —	— —	— —	— —	C_0 C_8
モード指定	W	77/3FDF ＊1	SC_1	SC_0	RL_1	RL_0	M_2	M_1	M_0	BCD

＊1：カウンタ 1 およびモード指定のアドレスは, 次のようになる

		カウンタ 1	モード指定
PC9801/E/F1,2,3/M2,3 PC98XA PC98XL PC98XL² PC98RL	 ハイレゾ・モード動作時 ハイレゾ・モード動作時 ハイレゾ・モード動作時	73H	77H
上記以外		3FDBH	77H または 3FDFH

〈図 2-24〉
タイマの I/O アドレス一覧

〈図 2-25〉 タイマ LSI8253 の周辺回路



〈図 2-26〉 カウンタの割り込み

	PC9801/E/F1,2,3/M2,3	左記以外	動作モード
カウンタ#0	インターバル・タイマ	インターバル・タイマ	モード 0
カウンタ#1	メモリ・リフレッシュ	スピーカ周波数設定	モード 3
カウンタ#2	RS-232-C	RS-232-C	モード 2

このピンが未使用ならば“1”が出力されますが，CLK 入力には 2.4576 MHz が使用されています．(p. 46 コラム参照)

CLK 入力は，拡張スロット用の「システム・クロック」を利用したものが使われます．CPU クロックに依存する機種もありますし，CPU クロックとシステム・クロックが別になっている機種もあり，CPU クロックだけでは判別がつかない場合も多いようです．

◈ タイマの用途

98 シリーズでは，3 組のカウンタを持っており，これらは図 2-26 に示すように，インターバル・タイマ，メモリ・リフレッシュ(旧機種)，ビープ(ブザー)用の音源，RS-232-C 用クロック発生に使用されています．

● カウンタ 0(モード#0)インターバル・タイマ

カウンタ 0 はインターバル・タイマで，指定した一定時間後に割り込みをかけることができます．システムは専用に使用していませんので，ユーザがアプリケーションの中で自由に利用できます．

一般的に使用されるモードは「#0」でカウント終了時での割り込みです．指定したカウントが終了したら割り込みがかかるモードで，カウント数を書き込んだ直後からカウントダウンを始め，“0”になったらソフトウェア割り込み(ベクタ#08H)がかかります．割り込みプログラムの中で，8253 に対して再度モード設定すると割り込み解除されます．モード「#0」は 1 回しか割り込みがかからないので，何度も定期的にかけたい場合はモード「#2」を使用します．

インターバル $n$ms の計算は前述の CLK 入力によって変わります．

▶システム・クロック 5/10 MHz 時

n×2457.6 に近い整数(最大 26.666 ms まで可)

▶システム・クロック 8 MHz 時

n×1996.8 に近い整数(最大 32.821 ms まで可)

● カウンタ 1(モード#3)ビープ音の周波数設定

カウンタ 1 は PC9801M 以前の機種ではメモリのリフレッシュに使用され，PC9801F では周期が 28.5 μs に設定されていました．ユーザがこれを変更すると動作が保証されなくなってしまいます．

PC9801VM 以降の機種では，ビープ(ブザー)用の音源に使用されています．プログラマブル・カウンタを使用していますから周波数は自由に変えられ，音楽の演奏等も可能です．

カウンタ 1 は機種によって動作は違いますから，プログラムを不用意に触って誤動作するのを防ぐために，ビープ音を設定する場合は，別アドレス(73H → 3FDBH，77H → 3FDFH)を設けています．もっとも，現在では PC9801M 以前の機種は，ほとんど使われていませんから問題ないでしょう．

ビープ音の周波数設定ではモード「#3」で方形波レート・ジェネレータを使用しています．これは，なるべくデューティ比 50 %の方形波を作ろうとするモードで，入力値が偶数ならデューティ比 50 %ですが，奇数の場合は，$n-1/2n$ となり，1 カウント分だけデューティ比 50 %からずれてしまいます．最悪の場合(n=5/n=3 は禁止)はデューティ比は 2：3 となりますが，可聴領域では「n」が大きくなるために，ほぼデューティ比 50 %となります．

ビープ音の周波数(f)設定の計算は以下のとおりです．

$n=F/f$

$F=2357600$(Hz)

(システム・クロック 5/10 MHz 時)

$F=1996800$(Hz)

(システム・クロック 8 MHz 時)

ビープ音の ON-OFF には，システム・ポート用の 8255・ポート C・ビット 4 を使用します．

ON：I/O アドレス 37H に 06H を書き込む．

OFF：I/O アドレス 37H に 07H を書き込む．

ビープ回路を図 2-27 に示します．

● カウンタ 2(モード#2)RS-232-C

カウンタ 2 は，RS-232-C 用のクロック発生用で，8251 の TxCLK/RxCLK に接続されています．

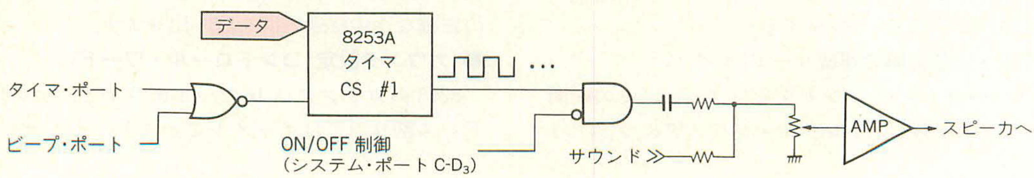
一般によく使用される非同期モードで使用する場合，ボーレートの 16 倍のクロックが必要になります．CLK 入力に 1.9968 MHz が使用されていると分周比の都合で 9600 bps を超える指定ができなくなります．

RS-232-C 用ボーレート設定は，ビープ音の周波数設定と同じ，モード「#3」で方形波レート・ジェネレータを使用しています．ボーレート算出方法は同期式，調歩同期式 1/16，調歩同期式 1/64 の方式別に図 2-28 にまとめました．

◈ LSI の使用方法

8253 にはハードウェア・リセットの端子はありませんし，初期化のためのコマンド等もありません．電源

〈図 2-27〉ビープ回路



投入以降，何かモードを指定するまでの動作は保証されません．

モードの指定は「コントロール・ワード」への書き込みで行われます．1バイト(8ビット)で，カウンタ・チャネル指定(0〜2)，動作モード指定(0〜5)等を行います．

チャネル別にカウンタは三つあり，それぞれアクセス・アドレスも別に割り当てられていますが，コントロール・ワードの書き込みは一つのアドレスしかありません．そのため，どのチャネルの制御を行うか指定します．

● モード設定(コントロール・ワード)

8253 のモードは，0〜5 まで 6 種類があります．

▶モード#0

指定した時間が経過すると OUT 端子が "H"(割り込みがかかる)になります．

コントロール・ワードを書くと OUT 端子が "L" になり，カウンタ・データをロードした直後にカウントダウンが始ります．カウントダウンが終了すると OUT 端子は "H" になります．

▶モード#1

指定された期間 "L" のパルスを出します．

コントロール・ワードを書くと "H" になります．カウンタ・データをロードした時点から，カウント数で指定した期間だけ "L" になります．カウントが終了して，ゲート端子が "L" になれば再びカウントし始めるのですが，PC98 ではゲート端子は使われていませんので，モード#0 と同じになります．

▶モード#2

指定した周期に一度パルス "L" を発生します．

コントロール・ワードを書くと "H" になります．データ・ロードするとカウントが開始され，カウントが終了する直前の 1 パルスだけ "L" になります．終了後は，再び初期値をロードしなおして繰り返されます．

カウント中にデータ・ロードしても，次のカウントダウンまで影響を与えません．

▶モード#3

モード#2 と同様ですが，"H" と "L" の期間が同じ(デューティ比 50 %)になるような出力を出します．ただし，カウンタ・データに「3」は使用できません．

〈図 2-28〉RS-232-C ボーレート設定

転送速度(ボー)	同期式		調歩同期式 1/16		調歩同期式 1/64	
	5/10 MHz	8 MHz	5/10 MHz	8 MHz	5/10 MHz	8 MHz
19200	128	使用不可	8	使用不可	使用不可	使用不可
9600	256	208	16	13	使用不可	使用不可
4800	512	416	32	26	8	使用不可
2400	1024	832	64	52	16	13
1200	2048	1664	128	104	32	26
600	4096	3328	256	208	64	52
300	8192	6656	512	416	128	104
150	16384	13312	1024	832	256	208
75	32768	26624	2048	1664	512	416

転送速度(ボー)	同期式	
	5/10 MHz	8 MHz
1200	128	104

▶モード#4

コントロール・ワードを書くと "H" になります．データ・ロードするとカウントが開始され，カウント終了後の 1 パルスだけ "L" になります．ゲート端子を "L" にするとカウントが停止しますが，PC98 シリーズではゲート端子の操作はできません．

▶モード#5

モード#4 と似ていますが，ゲート端子を "L" にするとカウントを終了し，再び新しくカウントを再開します．このモードも PC98 シリーズでは意味を持ちません．

98 シリーズでは，8253 のゲート端子はプルアップされているだけで使われていないために，ゲート端子を使用している，モード#1，#4，#5 は意味がありません．一般的に使用されるのは，#0，#2，#3 の 3 種類のようです．

● カウンタ・ラッチ(コントロール・ワード)

カウンタに対して指定する「リード/ライト・モード」というものがあります．これは，カウンタにデータを書いたり，読んだりするモードで，以下のようなものがあります．

#0：カウント・ラッチ・コマンド
#1：下位バイトのリード/ライト
#2：上位バイトのリード/ライト
#3：下位・上位の順で連続リード/ライト

カウント・ラッチ・コマンドを実行すると，その瞬間のカウンタの内容がストレージ・レジスタにラッチされ，正確なデータを読み出せます．この間はカウンタ自身の動作に影響を与えませんので，インタラプト等の正確な途中経過時間を読み出せます．

● **カウンタ設定(コントロール・ワード)**

8253のカウンタは16ビットあります．しかし，アドレス割り当てはチャネルごとに1バイト(8ビット)

〈図2-29〉コントロール・ワード

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC_1	SC_0	RL_1	RL_0	M_2	M_1	M_0	BCD

D_0:BCD

0	バイナリ・カウント（16桁）
1	BCDカウント（4桁）

カウンタをバイナリ・カウンタ（2進16桁）にするか，BCDカウンタ（10進4桁）にするか指定．BCD＝1でBCDカウンタになる．通常は「0」で使用

D_6～D_7：SC_0～SC_1

SC_0	SC_1	内容
0	0	カウンタ#0
0	1	カウンタ#1
1	0	カウンタ#2
1	1	マルチプル・ラッチ・コマンド

操作する，カウンタ#0～#2までの一つを選択する．タイマLSIにμPD71054(PC9801Fシリーズ以降)が載っている機種では，SC_0＝1,SC_1＝1でマルチプル・ラッチ・コマンドが使用可能．これは選択した複数のカウンタのその時点でのカウント・データ，プログラム状態，出力状態，カウント無効フラグ状態をすべて読み出せる．なおμPD8253相当の機種では無効になる

D_1～D_3:M_0～M_2

M_0	M_1	M_2	モード名	内容
0	0	0	モード#0	カウント終了時での割り込み
0	0	1	モード#1	ゲート端子リトリガブル・ワンショット
x	1	0	モード#2	レート・ジェネレータ
x	1	1	モード#3	方形波ジェネレータ
1	0	0	モード#4	ソフトウェア・トリガド・ストローブ
1	0	1	モード#5	ハードウェア・トリガド・ストローブ(リトリガブル)

カウンタ・モードを指定

D_4～D_5:RL_0～RL_1

RL_0	RL_1	内容
0	0	カウント・ラッチ・コマンド
0	1	下位バイトのみのリード/ライト
1	0	上位バイトのみのリード/ライト
1	1	下位・上位の順で連続リード/ライト

カウンタのリード/ライト・モードを指定

システム共有領域

システム・クロックや，8253に入力されるクロックの周波数を識別するには，プリンタ・ポートの42Hのビット5で認識します．しかし，ハイレゾ・モードやフル・セントロニクス・プリンタ・インターフェース等を使用した場合は，クロック周波数の認識ビットが異なります．

プログラムがハイレゾ・モードでも動作するように作るためには，ハイレゾ・モードかどうかを認識し，別のポートでクロック周波数を認識しなくてはなりません．

このクロック周波数は，ハードウェアのポートだけではなく，システム共通領域(0000：0501H)でも知ることができます．ビット7が“0”ならば，システム・クロックは5MHz系で，8253に入力されるクロックは2.4576MHzとなります．“1”の場合は8MHz系で，1.9968MHzのクロックがかかります．これを利用すると，ノーマル・モード，ハイレゾ・モードの区別なく認識することができます．

ただし，システム共通領域はシステム起動時のクロック周波数を示すもので，起動後にクロックを変更すると正しく認識できなくなってしまいます．

〈図2-A〉システム・クロックの識別

0000:0501H	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
	↑							

0＝5MHz系 (2.4576MHz)
1＝8MHz系 (1.9968MHz)
＊カッコ内は8253のクロック

〈図 2-30〉カレンダ時計のアドレス一覧

命令	I/O ポート・アドレス	READ/WRITE	データ							
			D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
セット・レジスタ	20	W	×	×	DI	CLK	STB	C_2	C_1	C_0
リード・データ	33	R	×	×	×	×	×	×	×	DO

×印：不定

しかありませんので，上位・下位を別々に読み書きすることになります．この指定は前述のリード/ライト・モードで行います．

リード/ライト・モードの#1や#2で，下位または上位のみを設定した場合は，その逆の桁には00Hが入ります．下位・上位の連続リード/ライトの場合は，連続して読み書きをしなくてはなりません．

◆ コントロール・ワード (77H/3FDFH/出力)

77Hでも，3FDFHでも同じポートが読み出せます．これはカウンタ#1の機種による用途の違いで誤動作が起きないようにするために分けられています．

図2-29にレジスタを示します．

◆ カウンタ・レジスタ (71H・73H/3FDFH・75H・入出力)

コントロール・ワード設定後に，カウンタのデータを読み書きする場合にアクセスします．

カウンタ#1は，機種によってアドレスが異なり，73H(PC9801M以前)と，3FDBH(PC9801VM以降)があります．これはカウンタ#1の機種による用途の違いで誤動作が起きないようにするために分けられています．

カレンダ時計

▶使用LSI
μPD1990C，μPD4990A
▶I/Oアドレス
20H
33H(システム・ポート)
▶使用割り込み
なし
▶初期設定命令
なし

図2-30にアドレス一覧を示します．

カレンダ時計用のLSIは機種によって異なるものが使用されています．PC9801/E/F/M/U/VF/VM/UV/XAにはμPD1990Cが使われ，それ以外の機種(PC9801VM後期モデルを含む)とEPSONのPCシリーズではμPD4990Aが使用されています．

カレンダ時計のLSIには3ビットのパラレル・コマンド・レジスタと，48(40)ビットの時刻データ・レジスタ，4ビットのシリアル・コマンド・レジスタがあります．CPUからアクセスするには，6ビット構成のセット・レジスタと，1ビットのリード・データを使用します．

セット・レジスタはI/Oアドレスの20H番地で，TTLのラッチにより行われます．リード・データは，システム・ポート用の8255のポートB(33H番地)ビット0を使用しています．

パラレル・コマンド・レジスタは，セット・レジスタのコマンド・コード(3ビット)にコマンドを書き込み，STBビットを0→1→0にすることで設定します．

時刻データ・レジスタとシリアル・コマンド・レジスタは，セット・レジスタのDIビットを使用して，シリアル形式で書き込みます．1ビット書き込むたびにCLKビットを1→0にして，これを48(40)回繰り返して設定します．

日付け・時刻のデータは，年(2桁)・月(1桁)・週(1桁)・日(2桁)・時(2桁)・分(2桁)・秒(2桁)から構成されます．1桁は4ビットでBCDですが，月は01H～0CHになります．

レジスタへの連続書き込みは，1～5μs(時間読み出しは20～40μs)と比較的時間がかかります．

◆ μPD1990とμPD4990の違い

μPD4990Aは，μPD1990の上位互換のLSIでテスト・モード(PC9801シリーズでは使用されていない)を除き，コンパチブルな動作をします．また，μPD4990Aでは「年」を扱うことが可能になり，「うるう年」の自動補正もやってくれます．BCDデータ2桁，8ビット分拡張されています．

μPD1990は四つのコマンドをパラレルで与え，日付けデータはシリアルで送受信します．μPD4990Aでは，パラレル・コマンドが八つに拡張され，さらに拡張されたシリアル形式でのコマンドも使え，合計16個のコマンドが使用可能になっています．

拡張されたコマンドはタイミング・パルス出力，インターバル出力関係で，インターバル・タイマのような機能を持っているのですが，PC9801シリーズでは使用されてなく，あまり意味を持ちません．

◆ 曜日の設定

〈図 2-31〉 セット・レジスタ

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
x	x	DI	CLK	STB	C_2	C_1	C_0

コマンド・コード
STB と共に用いてμPD4990 へパラレル・コマンドを与える.
シリアル・コマンド・モード時は, C_2, C_1, C_0=1, 1, 1 に保つ.

C_2	C_1	C_0	機　能
0	0	0	レジスタ・ホールド
0	0	1	レジスタ・シフト
0	1	0	タイム・セット＆カウンタ・ホールド
0	1	1	タイム・リード
1	0	0	TP=64Hz セット
1	0	1	TP=256Hz セット
1	1	0	TP=2048Hz セット
1	1	1	拡張モード・コマンド

（TP=64Hz セット～拡張モード・コマンド：μPD1990 では使用禁止）

ストローブ
μPD4990 へコマンドをセットするタイミングを与える

クロック
DO と共に用いて, シリアル・コマンドをμPD4990 へ時刻をセットしたり, 時刻を読み出すために用いる

入力データ（1 ビットごとに入力される）
CLK と共に用いてμPD4990 へシリアル・コマンドおよび時刻データをセットする

注:パラレル・コマンドで日付け,時刻を書き込むと,年のデータは壊れる

〈図 2-32〉 μPD4990 シリアル・コマンド

MSB　　LSB

→ | C'₃ | C'₂ | C'₁ | C'₀ | →

C'_3	C'_2	C'_1	C'_0	機　能
0	0	0	0	レジスタ・ホールド
0	0	0	1	レジスタ・シフト
0	0	1	0	タイム・セット/カウンタ・ホールド
0	0	1	1	タイム・リード
0	1	0	0	TP=64 Hz
0	1	0	1	TP=256 Hz
0	1	1	0	TP=2048 Hz
0	1	1	1	TP=4096 Hz
1	0	0	0	TP=1 sec インタラプト出力/カウンタ・リセット
1	0	0	1	TP=10 sec インタラプト出力/カウンタ・リセット
1	0	1	0	TP=30 sec インタラプト出力/カウンタ・リセット
1	0	1	1	TP=60 sec インタラプト出力/カウンタ・リセット
1	1	0	0	インタラプト出力リセット
1	1	0	1	インタラプト・タイマ・スタート
1	1	1	0	インタラプト・タイマ・ストップ
1	1	1	1	テスト・モード・セット

〈図 2-33〉 リード・レータ

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
x	x	x	x	x	x	x	DO

出力データ（1 ビットごとに出力される）
CLK と共に用いてμPD4990 から時刻を読み出す.

MS-DOS のシステム・コールの「日付けの取得」は, 曜日情報をカレンダ時計から得ずに, 年月日の情報から自分で計算します. また,「日付けの設定」でも, カレンダ時計に曜日情報を正確に書き込まずに, 常に 00H(日曜日)と設定してしまいます.

BIOS を使用した場合は, カレンダ時計が保持している曜日情報を, 正確に入出力しますので, MS-DOS で取得した曜日情報と食い違う可能性があります. 曜日情報を有効にする場合は, MS-DOS のシステム・コールを使用したほうが安全です.

◈ セット・レジスタ(20H/ライト)

D_0～D_5までのデータが, カレンダ時計 IC にラッチを通して接続されています.

▶ D_0～D_2：C_0～C_2(コマンド・コード)

コマンド入力, ファンクション・モードの選択等を指示します.

▶ D_3：STB(ストローブ入力)

ストローブ入力, データ・コマンドの入力時に使用する書き込みストローブ信号です.

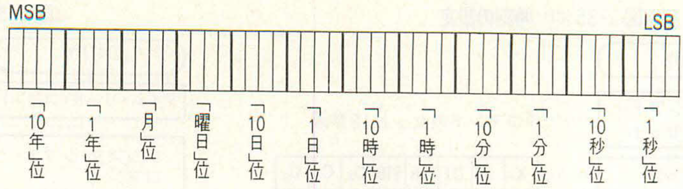
コマンド入力を確定してから 2 μs 後に, ストローブ入力を 0→1 に変化させ, 5 μs(40 μs：Time Read コマンド時のみ)後に, 1→0 にします. コマンド入力されたデータは, その後 2 μs 間変更してはなりません.

▶ D_4：CLK(クロック入力)

カレンダ時計 IC 内部のシフトレジスタのシフト・クロック入力です.

クロック入力の立ち上がり (0→1) でデータ (DI) を読み込みシフトします. クロック入力の前後の 2 μs

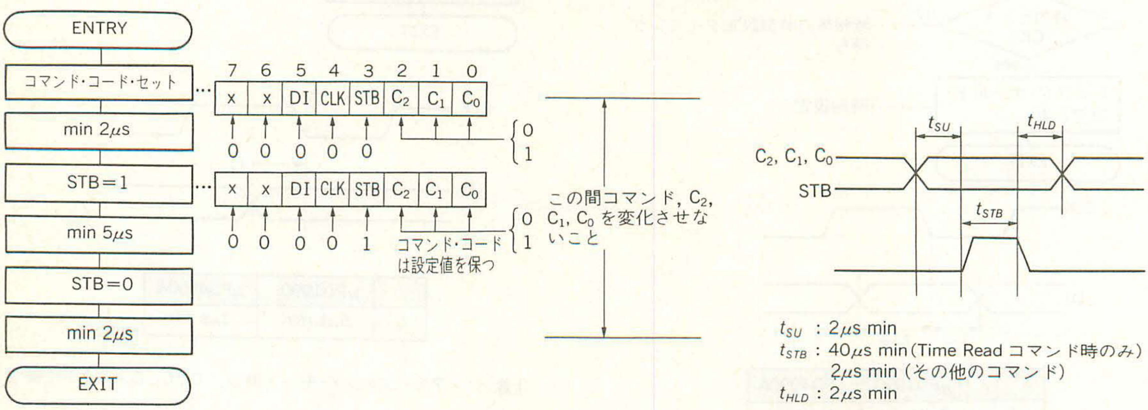
〈図 2-34〉
入出力データ形式



曜日	
日曜	0000
月曜	0001
火曜	0010
水曜	0011
木曜	0100
金曜	0101
土曜	0110

項目	形式	範囲
月	HEX DECIMAL	01H～0CH
曜	BCD	00H～06H
日	BCD	01H～31H
時	BCD	00H～23H
分	BCD	00H～59H
秒	BCD	00H～59H

〈図 2-35 (a)〉 パラレル・コマンドの使用方法



(4990 は 1 μs)間は DI を変化させてはなりません．

データ出力(DO)へ時刻データをシリアルで出力させるときにも使用されます．

▶ D_5：DI(データ入力)

シリアル・コマンドの入力や，時刻データの入力を行います．CLK 信号とタイミングをとってシリアルで入力します．

図 2-31 にセット・レジスタを，図 2-32 に μPD4990 のシリアル・コマンドを示します．

◈ リード・データ(33H/リード)

▶ D_0：DO(データ出力)

CLK とタイミングをとって，時刻データの出力に使います．

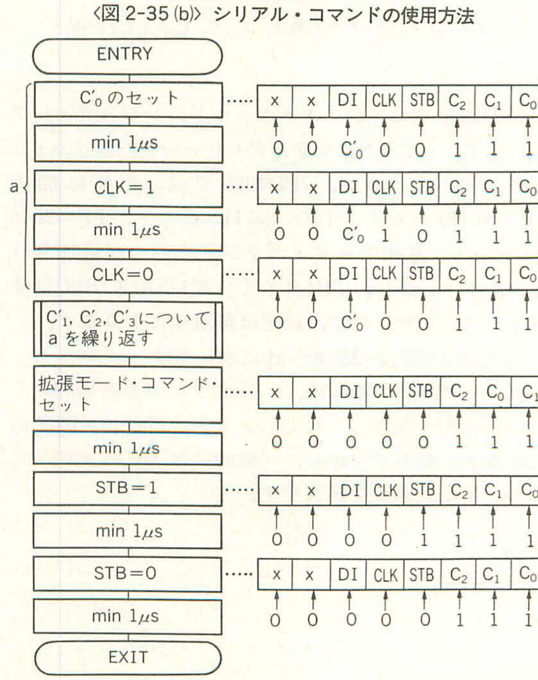
このポートはシステム・ポートの 8255 のポート B を使用しています．

図 2-33 にリード・データを図 2-34 (a)，(b)に入出力データ形式を示します．

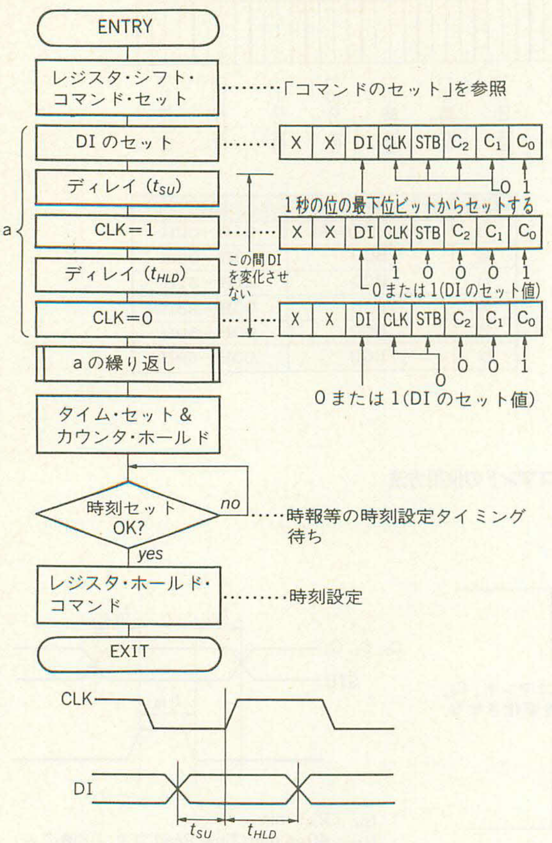
◈ カレンダ時計 IC のコマンド・コード

セット・レジスタの C_0～C_2に設定するデータです．

〈図 2-35 (b)〉 シリアル・コマンドの使用方法



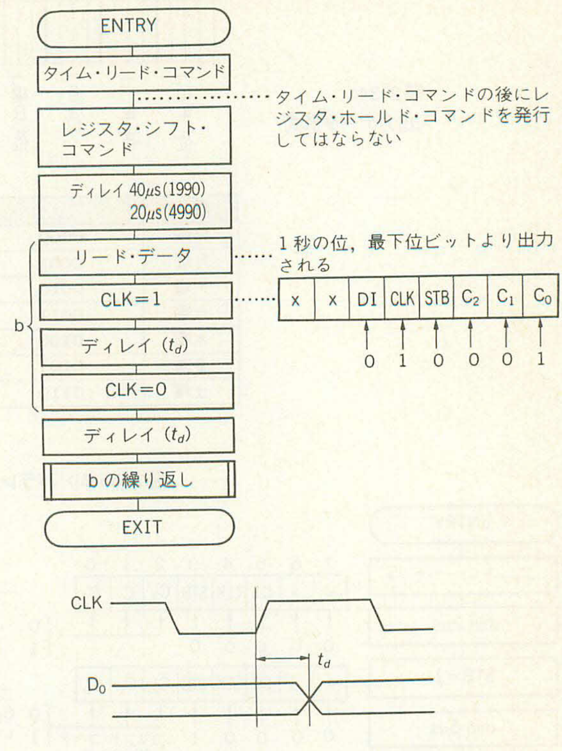
〈図 2-35 (c)〉 時刻の設定



	μPD1990	μPD4990A
t_{SU}	2μs min	1μs min
t_{HLD}	2μs min	1μs min

注:シリアル・コマンド・モード時は，C_2, C_1, C_0=1,1,1 に保つ

〈図 2-35 (d)〉 時刻の読み出し



	μPD1990	μPD4990A
t_d	5μs min	1μs min

注意:シリアル・コマンド・モード時は，C_2, C_1, C_0=1,1,1 に保つ

μPD1990 ではレジスタ・ホールド，レジスタ・シフト，タイム・セット&カウンタ・ホールド，タイム・リードの四つですが，μPD4990 では，それに加え TP=64 Hz セット，TP=256 Hz セット，TP=2048 Hz セット，拡張モード・コマンドの四つが追加されています．しかし，TP(タイミング・パルス)出力は使用されていませんので，設定は無意味になります．

使用方法を図 2-35 (a)〜(d)に示します．

カレンダ時計の操作は，ハードウェアを直接利用した場合の動作保証はされていません．日付け・時間の読み書きが基本ですから，一般的には BIOS を使って行ったほうが安全で確実です．

システム・ポート

▶使用 LSI

μPD8255A 相当

▶ I/O アドレス

31H，33H，35H，37H

▶使用割り込み

なし

▶初期設定命令

8255=92H

システム・ポートでは，ビープの ON/OFF や，カレンダ時計の読み出し，ディップ・スイッチの読み出し等を行うことができ，LSI にプログラマブル・パラレル・ポートの μPD8255A の相当品(以下，8255 と略す)を使用しています．

システム・ポートでの 8255 は，ポート A がモード 0/入力，ポート B がモード 0/入力，ポート C がモード 0/出力に初期設定されます．初期設定命令は 92H です．I/O アドレスの一覧を図 2-36 に，システム・

〈図 2-36〉システム・ポートのI/Oアドレス一覧

命　令	I/Oポート・アドレス	READ/WRITE	データ D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	備　考
ライト・モード	37	W	1	0	0	1	0	0	1	0	8255のモード・セット
ライト・ポートC	37	W	0	0	0	0	0	0	0	0/1	RXRE F/FのON/OFF 0：OFF，1：ON
	37	W	0	0	0	0	0	0	1	0/1	TXEE F/FのON/OFF 0：OFF，1：ON
	37	W	0	0	0	0	0	1	0	0/1	TXRE F/FのON/OFF 0：OFF，1：ON
	37	W	0	0	0	0	0	1	1	0/1	スピーカF/FのON/OFF 0：ON，1：OFF
	37	W	0	0	0	0	1	0	0	0/1	メモリ・チェックEnableのON/OFF 0：Disable，1：Enable
	37	W	0	0	0	0	1	0	1	0/1	$SHUT_1$のON/OFF
	37	W	0	0	0	0	1	1	0	0/1	プリンタのPSTB信号マスクF/FのON/OFF 0：イネーブル 1：マスク
	37	W	0	0	0	0	1	1	1	0/1	$SHUT_0$のON/OFF
ライト・ポートC	35	W	$SHUT_0$ (＊1)	PSTBM (＊2)	$SHUT_1$ (＊1)	MCKEN (＊3)	BUZ	TxRE	TxEE	RxRE	PORT Cの信号は本命令でもON/OFF可
リード・ポートC（診断用）	35	R	$SHUT_0$ (＊1)	PSTBM (＊2)	$SHUT_1$ (＊1)	MCKEN (＊3)	BUZ	TxRE	TxEE	RxRE	PORT Cの状態を読み取る
リード・ポートB	33	R	$\overline{CI}$	$\overline{CS}$	$\overline{CD}$	INT_3	CRTT	IMCK	EMCK	CDAT	PORT Bを通して各種信号を読み取る
リード・ポートA	31	R	$\overline{SW_8}$	$\overline{SW_7}$	$\overline{SW_6}$	$\overline{SW_5}$	$\overline{SW_4}$	$\overline{SW_3}$	$\overline{SW_2}$	$\overline{SW_1}$	PORT Aを通して各種スイッチ信号を読み取る

＊1：$SHUT_0$，$SHUT_1$は80286/386/486/Pentium搭載機種のみ
＊2：プリンタPST信号マスクF/F(PSTBM)は，PC9801では94HポートのD_4ビットを使用
＊3：メモリ・チェックEnable(MCKEN)は，PC9801U2では未使用

ポートの周辺回路を図2-37に示します．

◈ ポートA（31H/リード）

ポートAは入力に設定されていて，ディップ・スイッチSW_2の内容を読み出すことができます．それぞれのビットが意味する内容は機種によって若干変わります．ディップ・スイッチは，ハードウェアのものと，メニュ画面から設定するソフトウェアのもの，ハードウェアとソフトウェアの混在したものがあります．ポートA入力を図2-38に示します．

ディップ・スイッチは，ONにすると対応するポートAのビットが“0”になり，OFFになると“1”になります．

◈ ポートB（33H/リード）

ポートBは入力に設定されていて，主に，RS-232-Cの信号読み取り，カレンダ時計の読み出し，RAMのパリティ・エラーの原因表示等の読み出しができます．ポートB入力を図2-39に示します．

▶ D_0：

カレンダ時計からのデータをシリアル・データとして読み出すポートです．

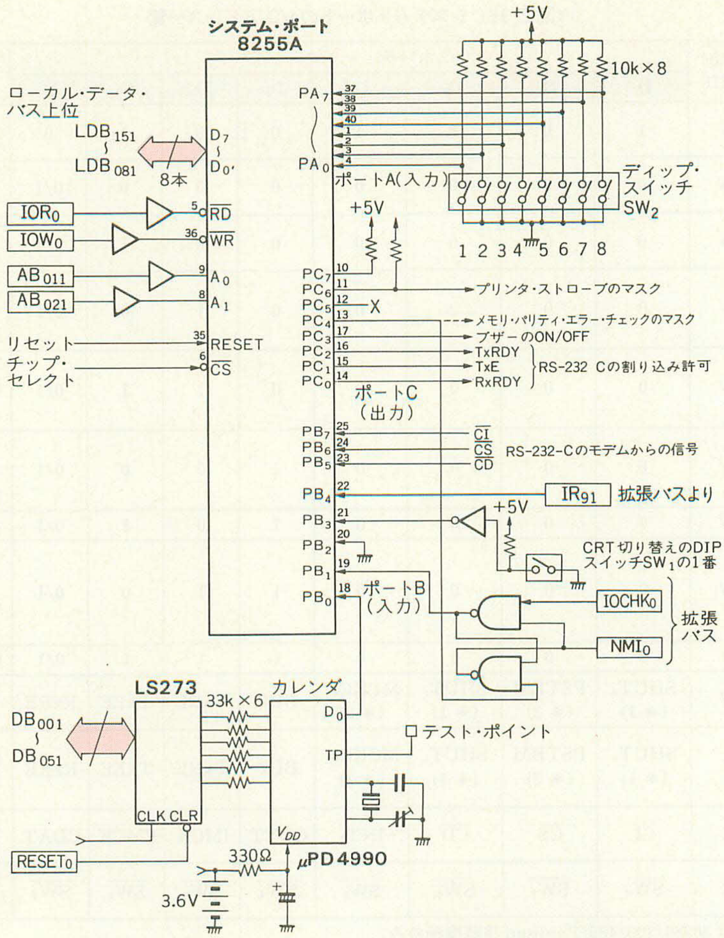
▶ D_1：外部RAMパリティ・エラー/D_2：内部RAMパリティ・エラー

RAMのパリティ・エラーが起きたときの原因区別用です．D_2は内蔵RAM，専用増設RAM用で機種によっては未使用なものもあります．D_1は「拡張RAM用」です．“0”でエラーなし，“1”でエラー発生です．RAMのパリティ・エラーが発生すると，ハードウェア割り込みであるNMI割り込み（ベクタ#2）がかかります．この割り込みプログラムはエラー原因を知るために使います．

▶ D_3：CRTタイプ

1＝高解像度，0＝標準解像度を選びます．ディッ

〈図 2-37〉
システム・ポートとカレンダ



〈図 2-38〉 ポート A 入力

データ・ビット	信号名	備　　考
D_7	SW_8	ON：GDC 5 MHz，OFF：GDC 2.5 MHz　(＊1)
D_6	SW_7	未使用
D_5	SW_6	ON：固定ディスク切り離し，OFF：接続　(＊2)
D_4	SW_5	ON：メモリ・スイッチ保持，OFF：メモリ・スイッチ初期化
D_3	SW_4	ON：25 行/画面，OFF：20 行/画面
D_2	SW_3	ON：80 文字/行，OFF：40 文字/行
D_1	SW_2	ON：ターミナル・モード，OFF：BASIC
D_0	SW_1	常に OFF

＊1：PC9801/E/F1,2,3/M2,3/U2/VF2/VM0,2,4 では未使用
＊2：固定ディスク内蔵モデルのみ，その他では未使用
＊3：PC9801N ではディップ・スイッチが 1 個となったため，D_4，D_7ビットのみ有意となる

プ・スイッチの SW_{1-1}が読み取れます．

▶ D_4：INT_3

ハード・ディスク割り込み信号です．

▶ D_5：CD/D_6：CS(CTS)/D_7：CI

RS-232-C 用に使用されている 8251 では読み取ることのできない，CI，(CTS)CS，CD，の信号を読み取ることができます．RS-232-C の各信号が，＋V(OFF)の時や，開放されている時に対応するビットが "1" になります．－V(ON)の時は "0" になります．

◈ ポート C(35H/ライト)

ポート C は出力に設定されていて，RS-232-C の割り込み制御，ビープの ON/OFF，プリンタの PSTB

〈図 2-39〉ポートB入力

データ・ビット	信　号　名	備　　考
D_7	$\overline{CI}$ (RS-232-C)	RS-232-C　CI 信号
D_6	$\overline{CS}$ (RS-232-C)	RS-232-C　CS 信号
D_5	$\overline{CD}$ (RS-232-C)	RS-232-C　CD 信号
D_4	INT_3	固定ディスク　INT 信号
D_3	CRT TYPE	1：高解像度，0：標準解像度
D_2	内部 RAM パリティ・エラー	標準 RAM のパリティ・エラー
D_1	外部 RAM パリティ・エラー	拡張 RAM のパリティ・エラー
D_0	カレンダ時計の読み出しデータ	

〈図 2-40〉ポートC出力

データ・ビット	信　号　名	備　　考
D_7	$SHUT_0$	
D_6	PSTB マスク	プリンタの $\overline{PSTB}$ 信号のマスク
D_5	$SHUT_1$	
D_4	メモリ・チェック Enable	1：エラー登録する＊，0：エラー登録しない
D_3	ブザー制御 F/F	1：ブザー停止，0：鳴動
D_2	TXR Enable F/F	RS-232-C の TXRDY による割り込みの Enable
D_1	TXE Enable F/F	RS-232-C の TXEMPTY による割り込みの Enable
D_0	RXR Enable F/F	RS-232-C の RXRDY による割り込みの Enable

信号の制御等に使われています．このポートCはバイト単位(8ビット1組)でなく，ビット単位で制御できる「ビット・モード」を使って操作するのが便利です．ポートC出力を図 2-40 に示します．

▶ D_0：RxRDY 割り込みイネーブル/D_1：TxEMPT 割り込みイネーブル/D_2：TxRDY 割り込みイネーブル

RS-232-C 用割り込みのマスク用です．“1”で割り込み禁止，“0”で割り込み可能になります．

▶ D_3：ビープの ON-OFF

ビープ音の ON-OFF 用です．“0”で鳴り(ON)，“1”で停止(OFF)します．

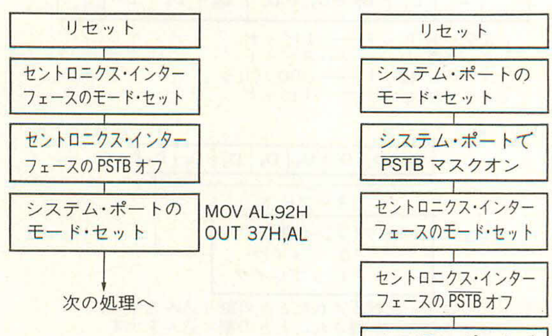
▶ D_4：パリティ・エラーのイネーブル

パリティ・エラー用の NMI 割り込みの制御で，“0”で禁止，“1”で許可です．

▶ D_6：PSTB マスク

プリンタ・ストローブ信号のマスク制御用です．8255 は出力モード設定直後はすべて“0”となるために，プリンタ用 8255 とシステム・ポート用 8255 を設定する際に設定順序を考えないと，プリンタが誤動作する可能性があります．モード設定フローを図 2-41 に示します．

〈図 2-41〉モード設定



(a)　マスク・セット不要

(b)　マスク・セット必要

▶ D_5：$SHUT_1$/D_7：$SHUT_0$

80286 以上の CPU を使用する際に，ハードウェア・リセットがかかったか，プロテクト・モードからの復帰なのかを判別するポートです．CPU に 8086 や V30 を使用した機種では未使用です．

80286 では，プロテクト・モードから，リアル・モードへ復帰する際に，CPU に対してリセットをかけなくてはならないため，ソフトウェアから CPU のみをリセットすることができます．このポートを読むことで，ハードウェア・リセットがかかったのか，CPU だけリセットされたのか判別することができます．

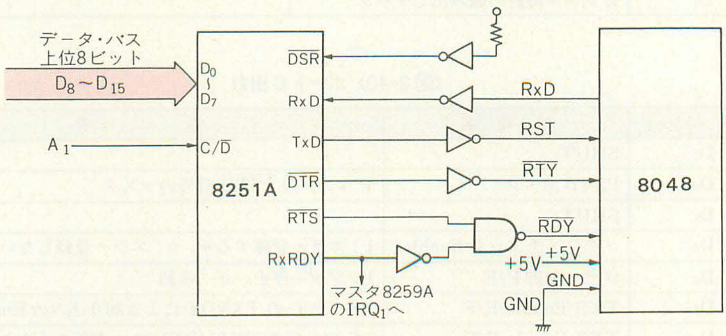
仕組みは，ハードウェア・リセットがかかると，8255 も初期化され，ポートCは入力モードになり，ビット 5，7 の入力をプルアップしておけば“1”が読み出せます．その後，プログラムが 8255 のポートCを「出力」として設定すれば，ビット 5，7 は“0”になりますので区別ができます．

$SHUT_0$＝0：プロテクト・モードから復帰

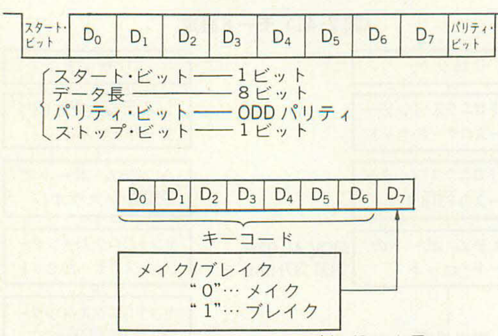
〈図 2-42〉キーボード・インターフェースの I/O アドレス

命　令	I/O ポート・アドレス	READ/WRITE	機　能
モード/コマンド・ライト	43	W	モード・セット コマンド・セット
データ・リード	41	R	μPD8251A にロードされた1バイト・データを読み出す
ステータス・リード	43	R	μPD8251A のステータスを読み出す

〈図 2-44〉キーボード・インターフェースの回路



〈図 2-43〉シリアル・データのフォーマット(上)，
データのフォーマット



注：メイクはキーが押下されたときの割り込みを示し，
ブレイクはキーが離されたときの割り込みを示す

$SHUT_0$=1：ハードウェア・リセット

起動時に BIOS がポート 35H から $SHUT_0$を読み，0 であればプロテクト・モードからの復帰(ソフトウェア・リセット)と判別し，SP ← 0000：0404H/SS ← 0000：0406H として「FAR RET」します．1 であれば，電源投入時かリセット・スイッチが押された(ハードウェア・リセット)と判別します．

例外として，EPSON の PC286 では，プロテクト・モードからの復帰判別に，0C03H のポートを使用します．読み出したデータが 50H(大文字英字の「P」)であれば，プロテクト・モードからの復帰と判別し，SP ← 0000：0400H/SS ← 0000：0402H として「FAR RET」します．

キーボード・インターフェース

▶使用 LSI

μPD8251A 相当(本体側)

μPD8048 相当(キーボード側)

▶I/O アドレス

41H，43H

▶使用割り込み

IR_1(ベクタ#09H)

▶初期設定命令

8251＝5EH

図 2-42 に I/O アドレス一覧を示します．

キーボードには，セパレート・タイプ(本体とケーブルにより接続される)と，ノート・パソコンのように本体と一体型のものがあります．さらに，「CAPS」，「カナ」のキーがメカニカル・ロックされるものと，ソフトウェア制御により LED 等で表示されるものの 2 種類があります．

キーの数は，79，84，100，101，106，107 の 6 種類があります．98LT は 79 個，それ以外のノート・ラップトップでは 84 個，CPU に 8086 を持つ古い機種は 100 個，CPU に V30 や 80286 を持つ古い機種では「NFER」キーが増えて 101 個，比較的新しいものはファンクション・キー「vf・1〜vf・5」が 5 個増えて 106 個で最近の標準となっています．ハイレゾ・モードを持つタイプでは「HOME」キーが独立して 107 個あります．

8255について

8255は，8ビットの汎用パラレル入出力ポートを3組(合計24ビット)持ったLSIです．ポートは，「ポートA」，「ポートB」，「ポートC(上位4ビット)」，「ポートC(下位ビット)」の四つに分けられ，プログラムから自由に「入力」，「出力」へ設定できます．また，「ポートC」は，ビット単位で内容を可変できる「ビット操作モード」を持ちます．

モード設定では，「ポートA」，「ポートC(上位4ビット)」をAグループ，「ポートB」，「ポートC(下位ビット)」をBグループと区別します．

I/Oアドレスは，「ポートA，ポートB，ポートC」と「コントロール・レジスタ」の四つがあります．コントロール・レジスタで，各ポートの「入力/出力の指定」と「モード設定」，そして「ビット操作モード」を行います．

● モード#0

8255には三つのモードがあります．モード0は，単純な入出力モードで，三つのポートのすべてが汎用入出力ポートになります．出力設定ではCPUがライトしたデータがそのままポートに現れ，CPUがリードすると前にライトしたデータが読めます．入力設定にした場合は，ポートの内容をそのままCPUが読み取ることができます．PC9801シリーズの「ノーマル・モード」では，すべてモード0で使われています．

● モード#1

「入力」，「出力」の片方向のハンドシェイクができます．ハンドシェイクとは，CPUがポートを読み出すまで，ハード的に相手からのデータ送出を停止させられる機能で，同時に割り込み処理も可能になります．プリンタ等のCPUに対して著しく速度が違う周辺機器を制御する場合に，割り込み等での処理が可能になり便利です．ハイレゾ・モードでのプリンタ出力に使われています．

モード1では，ポートA/Bを使用することができ，ポートCはハンドシェイクを実現させるために，1組で3個(A/B両方なら6個)を制御線に使います．残ったポートCは，モード0の汎用ポートとして使用することができます．

● モード#2

入出力の双方向ハンドシェイクができるモードですが，ポートAだけしか使用できません．この場合はポートCの5個が制御線に使用されます．PC9801シリーズでは使用されないモードです．

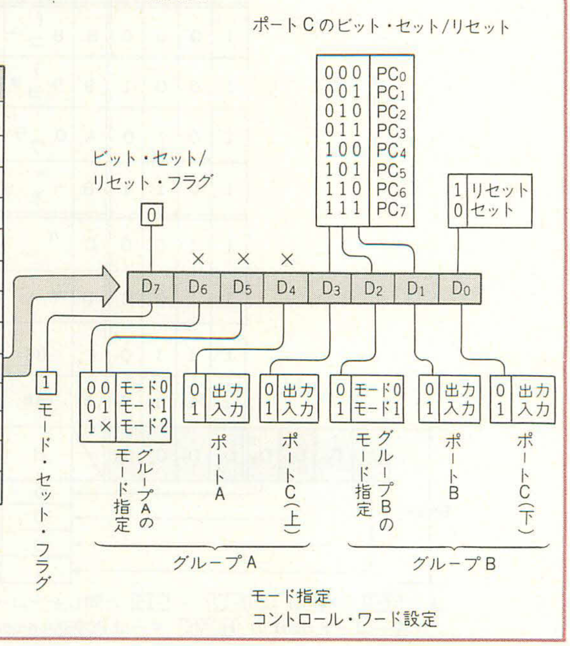
● 8255の初期設定

図2-Bはコントロール・レジスタの設定方法です．最上位ビットが“1”の場合は，8255の初期設定になり，いつでも自由に設定が可能です．最上位ビットが“0”の場合は，「ビット操作モード」になります．ポートCは常にモード0に設定されますが，グループA/Bで，モード1/2に設定する場合は，その一部が制御線として使われます．

〈図2-B〉8255Aのコントロール・ワードと設定例

ポート指定

	$\overline{CS}$	A_1	A_0	$\overline{WR}$	$\overline{RD}$	入力動作 (READ)
IN命令で，A, B, Cの各ポートのデータを読み取れる	L	L	L	H	L	ポートA→データ・バス
	L	L	H	H	L	ポートB→データ・バス
	L	H	L	H	L	ポートC→データ・バス
OUT命令で，A, B, Cの各ポートへデータを出力できる						出力動作 (WRITE)
	L	L	L	L	H	データ・バス→ポートA
	L	L	H	L	H	データ・バス→ポートB
	L	H	L	L	H	データ・バス→ポートC
モード設定のためのコマンドを書き込む．このモード設定で各ポートの状態が決まる．D_7=0のときは，ポートCの各ビットのON/OFFの制御ができる						コントロール
	L	H	H	L	H	データ・バス→コントロール・レジスタ
						機能なし
	H	×	×	×	×	データ・バス→3ステート
	L	H	H	H	L	イリーガル状態
	L	×	×	H	H	データ・バス→3ステート



◈ キーボードと本体とのデータのやりとり

キーボードと本体との間は，シリアル方式でデータ転送されています．4本の制御線・データ線と，+5Vの電源ライン，グラウンドの合計6本でつながれており，シリアル通信にはRS-232-Cと同じμPD8251A相当(以下8251と略す)が使用されています．キーボード側はμPD8048相当のワンチップCPUで制御されています．

シリアル通信方式は，19200 bps，8ビット，スタート・ビット1，ストップ・ビット1，パリティ奇数，調歩同期式，TTLレベルになっています．データの発生は，キーボードが押されたとき(Make)と離されたとき(Break)で，0.5秒以上押してリピート機能が動作しているときは，MakeとBreakが連続してキーボード側で発生されます．

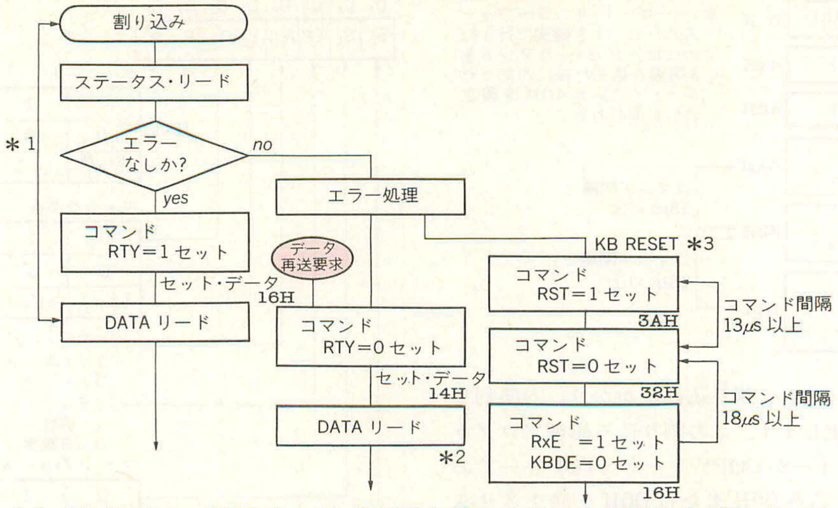
シリアル・データのフォーマットおよびデータのフォーマットは，図2-43のようになります．キーボード(8048)から送られてくるデータは8ビット構成ですが，最上位ビット(D_7)は，Make/Breakの情報を表

〈図2-45〉キー・コード一覧

Make D_7						0	0	0	0	0	0	0	0
Make D_6						0	0	0	0	1	1	1	1
Make D_5						0	0	1	1	0	0	1	1
Make D_4						0	1	0	1	0	1	0	1
D_7 D_6 D_5 D_4	D_3	D_2	D_1	D_0		0	1	2	3	4	5	6	7
	0	0	0	0	0	ESC	Q タ	F ハ	, < ネ 、	−	・	STOP	SHIFT
	0	0	0	1	1	1 ! ヌ	W テ	G キ	. > ル 。	/	NFER	COPY	CAPS
	0	0	1	0	2	2 " フ	E イ ィ	H ク	/ ? メ ・	7	f·11	f·1	カナ
	0	0	1	1	3	3 # ア ァ	R ス	J マ	− ロ	8	f·12	f·2	GRPH
	0	1	0	0	4	4 $ ウ ゥ	T カ	K ノ	△	9	f·13	f·3	CTRL
	0	1	0	1	5	5 % エ ェ	Y ン	L リ	XFER	*	f·14	f·4	
	0	1	1	0	6	6 & オ ォ	U ナ	; + レ	ROLL UP	4	f·15	f·5	
	0	1	1	1	7	7 ' ヤ ャ	I ニ	: * ケ	ROLL DOWN	5		f·6	
	1	0	0	0	8	8 (ユ ュ	O ラ	] } ム 」	INS	6		f·7	
	1	0	0	1	9	9) ヨ ョ	P セ	Z ツ ッ	DEL	+		f·8	
	1	0	1	0	A	0 ワ ヲ	@ ~ ゛	X サ	↑	1		f·9	
	1	0	1	1	B	− = ホ	[{ ゜ 「	C ソ	←	2		f·10	
	1	1	0	0	C	^ ` ヘ	⏎	V ヒ	→	3			
	1	1	0	1	D	¥ ¦ _	A チ	B コ	↓	=			
	1	1	1	0	E	BS	S ト	N ミ	HOME CLR	0	HOME		
	1	1	1	1	F	TAB	D シ	M モ	HELP	,			
D_7 D_6 D_5 D_4	D_3	D_2	D_1	D_0		8	9	A	B	C	D	E	F
Break D_4						0	1	0	1	0	1	0	1
Break D_5						0	0	1	1	0	0	1	1
Break D_6						0	0	0	0	1	1	1	1
Break D_7						1	1	1	1	1	1	1	1

注：vf·1 ～ vf·5 は f·11 ～ f·15 と同じキーコードを発生する
キーコード5EHの HOME キーはPC98XA model 1, 2, 3/11, 21, 31/XL model 1, 2, 4/XL²/RLのみにある

〈図 2-46〉キーボード割り込み処理



＊1：割り込みから，DATA リードまでの時間が 37μs 以上になる必要がある（KB に対して DRY＝1 のパルス幅が 37μs 以上必要となる）
＊2：ステータスに異常があり，RTY＝0 をセットした場合でも，必ず DATA を引き取ること
＊3：数回リトライを行ってもステータスに異常がある場合には，KB RESET を行う

し，キーコードは 7 ビットになっています．

制御線は，RxD(RxD/括弧内は 8251 の信号線名)でキーボードからのデータを受けます．RDY(RxRDY&RTS)はキーボードへデータ受け取り準備完了を示します．RTY(DTR)はキーボードへ前回受けたデータの再送信要求信号です．RST(TxD)はキーボードの CPU を初期化する信号です．8251 の TxD は本来はデータ送信用に使用している端子なのですが，ブレーク信号発生コマンドを実行すると，TxD が“L”レベルになることを利用して，汎用出力端子として使用しています．

図 2-44 にキーボード・インターフェースの回路を示します．

◈ キーボードのソフトウェア制御

ソフトウェア制御の可能なキーボード(主に PC9801RA 以降)では，本体側からキーボードの制御ができ，「キーボードのリセット」，「キーボードのタイプ認識」，「LED の制御」等が可能です．また，「CAPS」「カナ」の状態は本体の電源を消しても記憶されています．

ソフトウェア制御は，RST(TxD)用の制御線を使ってコマンド(データ)を送ります．本来は初期化用の制御線ですから，コマンド発生にはいくつかの条件があります．RST 信号発生直後(0 ms)～3 ms と，10～50 ms の間は，コマンド発生禁止です．また，キーボードから本体側へデータを送っている途中は，データをすべて送り終わるまでコマンドは処理されません．

ソフトウェア制御の可能なキーボードかどうかの判別は，システム共通領域の 0000：0481H のビット 6 で調べられます．“1”の場合がソフトウェア制御の可能なキーボードです．

古いタイプのキーボードでは，本体からキーボード・ケーブルを抜いて，再度差しても，キーボードの CPU がリセットされずに，本体をリセットするまで使用不能でしたが，ソウトウェア制御ができるキーボードでは，脱着してもキーボードが使用できなくなることはありません．EPSON-PC シリーズでは，メカニカル・ロックのキーボードを脱着しても使用不能にはなりません．

ソフトウェア制御のキーボードを古いタイプのキーボードが付く機種につなげたり，その反対をしてもキーボードとして動作します．しかし，電源を切ると，キーボードの状態と本体のキーの記憶状況(古いタイプの機種では記憶されない)とが合わなくなる場合があります．電源を入れるたびに「CAPS」，「カナ」のキーを空押しすれば，状態は一致できます．

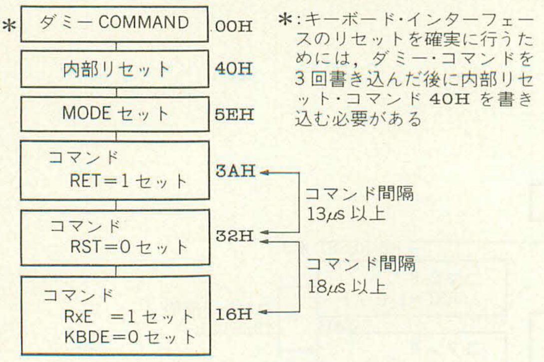
◈ キーボード割り込みの処理

キーボードから送られるシリアル・データ(キー・コード)は，図 2-45 のように 107 個(SHIFT キーとリターン・キーが二つずつあるので 105 種類)のキーを 7 ビットで表します．そのキーが押されたとき(Make)に(0)，離されたときに(Break)に(1)のデータを 1 ビットで表し，合計 8 ビットのデータになります．

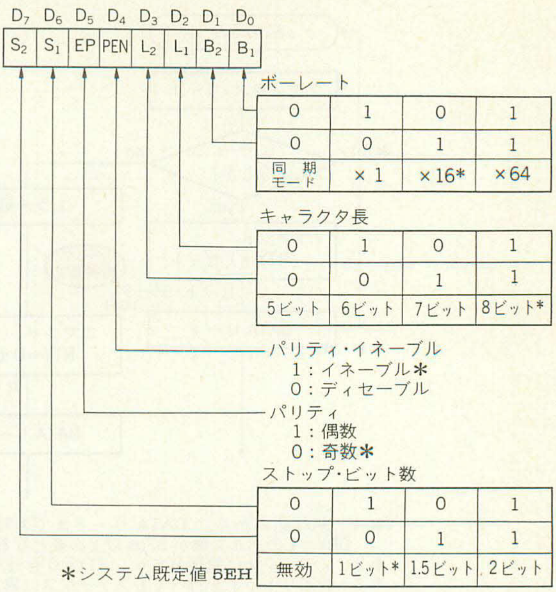
ここで発生するデータは，JIS コードとは無関係で，キーボードのキーそのものに与えられたコード番号になっていて，SHIFT キーやコントロール・キー等もデータとして得られます．

8251 がキー・コードを受信すると，割り込みコント

〈図 2-47〉 8251 の初期設定



〈図 2-48〉 モード・ライト



ローラ(8259)の IRQ_1 に割り込みがかかり，内部割り込み 09H が発生します．この割り込み処理プログラムは，押されたキーが COPY キーか STOP キーであれば，内部割り込み 05H または 06H を発生させます．受信したキー・コードからキー入力状態テーブル(0000：052AH～0539H)を作成し，この内容からシフト状態を考慮したうえで JIS コードへコード変換します．

変換された JIS コードと，元のキー・コードを合わせた2バイトを，0000：0502H～0521H の BIOS のキー・バッファへ格納し，同時に各ポインタを書き換え，処理は終了します．

EPSON の PC シリーズの一部の PC の機種では，キーボード割り込みの処理が，NEC の 98 シリーズと違うものもありますので，注意が必要です．

図 2-46 にキーボード割り込み処理フローを示します．

◈ キーボード・インターフェースの初期化と制御

キーボード・インターフェース用の 8251 の初期設定データは 5EH です．8251 の初期設定後は，キーボードをリセットするために「ブレーク信号」を送り，キーボード送信許可(RTS=0)，再送信要求有効(DTR=0)，受信割り込み許可(RxE=1)，送信禁止(TxEN=0)，エラー・リセット(ER=1)にして，キーボードからのデータを待ちます．

8251 の I/O アドレスは二つあり，モード・コマンド書き込み(設定)/ステータス読み出しポートと，データの読み書きポートです．モード・コマンド設定ポートは一つのアドレスを共用しており，モード設定(初期化データ 5EH を書き込む)は 8251 をリセットした直後に一度だけ設定でき，二度目からはコマンド設定ポートになります．8251 のリセットはコマンド設定(ソフトウェア)から行うので，決められた手順どおりに「リセット→モード設定(図 2-47)」する必要があります．

◈ 8251 のレジスタ

8251 の各レジスタの内容は以下のとおりです．

● モード設定(43H/ライト)

キーボード・インターフェースでは，19200 bps，8 ビット，スタート・ビット 1，ストップ・ビット 1，パリティ奇数，調歩同期式に固定されています．8251 の TxC/RxC には，307.2 kHz が供給されているのでボーレート指定は「×16」となり，初期設定データは 5EH となります(図 2-48)．

● コマンド設定(43H/ライト)

図 2-49 はコマンドの設定です．

▶ D_0：TxEN

送信の許可/禁止を指示します．送信禁止(TxEN=0)にすると，その時点で書き込まれているデータをすべて送出してから送信を停止します．

▶ D_1：RTY(DTR)

受信データにエラーがあった場合に，再送信要求を指示します．RTY=0 で再送信要求です．8251 の汎用出力ポート(DTR)の制御用です．

▶ D_2：RxEN

受信の許可/禁止を指示します．RxEN=0 で受信禁止です．

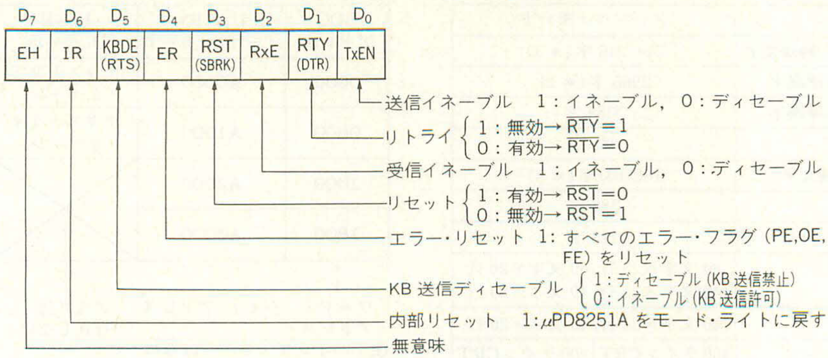
▶ D_3：SBRK

キーボードのリセットを行います．リセット時には 13 μs だけ SBRK=1 とします．本来ブレーク信号送出用で，SBRK=1 のときに 8251 の TxD 出力を "L" レベルにします．

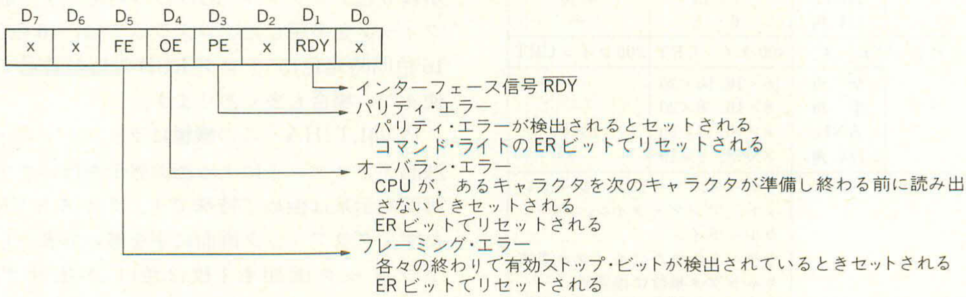
▶ D_4：ER(ECL)

8251 のエラー・ステータス(PE，OE，FE)のクリア

〈図 2-49〉 コマンド・ライト



〈図 2-50〉 ステータス・リード



を行います．ER=1 でエラーはクリアされます．

▶ D_5：KBDE(RTS)

キーボード送信の許可/禁止を指示します．KBDE=0 で送信許可です．8251 の汎用出力ポート (RTS) の制御用です．

▶ D_6：IR(SRES)

8251 をソフトウェア・リセットさせます．IR=1 でリセットし，モード設定待ち状態になります．

▶ D_7：EH

8251 の同期モード用で，キーボード・インターフェースでは使用しません．

◈ ステータス読み出し (43H/リード)

図 2-50 はステータス・リードです．

▶ D_0：TxRDY

送信データ・バッファ状態を示します．TxRDY=0 でバッファにデータがあることを示し，この状態ではデータを送出できません．

▶ D_1：RxRDY

RxRDY=1 でデータを受信したことを示します．

キーボード・インターフェースでは，通常は割り込みで制御されますので，このポートを監視する必要はありません．

▶ D_2：TxEMP

送信データ・バッファ (第 2 バッファ) とトランスミッタ内の送信バッファ (第 1 バッファ) が共に空であることを示します．

▶ D_3：PE

パリティ・エラーの発生を示します．エラーがあれば "1"，なければ "0" になります．エラーが発生しても 8251 の動作は停止しません．ER=1 でエラーはクリアされます．

▶ D_4：OE(OVE)

オーバラン・エラーの発生を示します．CPU が受信データの読み出しに遅れたときに "1" になります．

エラーが発生しても 8251 の動作は停止しません．ER=1 でエラーはクリアされます．

▶ D_5：FE

フレーミング・エラーの発生を示します．ストップ・ビットが検出されなかったときに "1" になります．エラーが発生しても 8251 の動作は停止しません．ER=1 でエラーはクリアされます．

▶ D_6：SYNC/BRK

調歩同期モードでは，ブレーク信号 (RxD が 2 キャラクタ以上の時間 "0" になった場合) を受信したときに "1" になります．キーボード・インターフェースでは使われません．

▶ D_6：DSR

汎用入力ポートの DSR の状態を示します．キーボード・インターフェースでは使われません．

〈図2-51〉 PC98シリーズのテキスト表示の機能概要

			ノーマル・モード	
表示文字種	ANK文字，特殊文字		244/246字(＊1)	
	JIS第1水準漢字		2965字(＊2)	
	JIS第2水準漢字		3384字	
	非漢字		885字	
	ユーザ定義文字		188(63)字(＊3)	
	拡張漢字		388字	
	1/4角文字(＊4)		213字	
表示文字容量	ANK		80文字×25行，80文字×20行 40文字×25行，40文字×20行	
	全角文字		40文字×25行，40文字×20行	
表示文字構成 ドット数，横×縦	レターフェース		400ラインCRT	200ラインCRT
		全角	15×16	—
		半角	7×16	—
		ANK	7×13	6×8
		1/4角	6×8	—
	ボディーフェース		400ラインCRT	200ラインCRT
		全角	16×16，16×20	—
		半角	8×16，8×20	—
		ANK	8×16，8×20	8×8
		1/4角	8×8，8×10	—
アトリビュート			リバース，ブリンク，シークレット，アンダ・ライン，バーチカル・ライン カラー8色またはモノクロ濃淡 キャラクタ単位に指定可	
VRAM			テキスト表示用4KB，日本語表示用4KB，アトリビュート4KB計12KB CPUにより直接READ/WRITE GDCによる描画機能なし パリティ・ビットなし	

＊1：PC9801/E/F1，2，3/M2，3/U2/VF2/VM0，2，4/UV2には，バック・クォートとバック・スラッシュがない
＊2：日本語表示には，専用高解像度ディスプレイ(400ラインCRT)が必要
＊3：PC9801では，ユーザ定義文字は不可．PC9801E/F1，2，3/M2，3/U2では，ユーザ定義文字は63文字
＊4：1/4角文字はグラフィック画面にのみ表示可能
注：98NOTE，PC9801BA，BX，PC9821，Ap，As，Ae，Ce，Af，PC9801Pは専用高解像度ディスプレイ固定である

〈図2-52〉 テキストVRAMのメモリ空間

GDCアドレス	CPUアドレス ノーマル・モード	HIGH D_{15}……D_8	LOW D_7……D_0
0000	A0000	テキスト文字 1ページ	テキスト文字 1ページ
0800	A1000	テキスト文字 2ページ	テキスト文字 2ページ
1000	A2000	╳	アトリビュート 1ページ
1800	A3000	╳	アトリビュート 2ページ

↑ワード・アドレス　↑バイト・アドレス　↑メモリは存在しない

注：ハイレゾ・モードは省略

CRTディスプレイ

◈ 機種による仕様の違い

PC98シリーズには実に様々な機種があり，また機種によってCRTディスプレイの機能に違いがあります．基本的には，80桁×25行の漢字を表示できるテキスト表示と，640×400×16色のグラフィック表示を2枚ずつサポートしています．

PC9801/U…これらの機種ではグラフィックVRAMが1組しかありません．最近のプログラムでは，VRAMが2組あるものを前提に作られているものも多く，異端的な機種になっています．

PC9801/E/F/M…これらの機種のグラフィック表示は8色ディジタルRGBのみ対応です．最近のグラフィックを多用したプログラムでは，16色(4096色中16色同時発色)アナログRGB対応が普通ですから，使えない場合も多くあります．

PC98LT/HA…この機種はラップトップ・タイプで液晶ディスプレイによる画面表示を行います．また，画面表示系は極めて特殊です．テキストVRAMを持たず，グラフィック画面に字を書いて表示します．グラフィック画面も1枚(2色)しか出せず，従来のPC98シリーズとは大きく異なります．

その他のラップトップ，ノート系のコンピュータでは，通常のPC98シリーズと同じ仕様になっています．液晶ディスプレイによる制限で画面上では白黒8階調しか表示できないものもありますが(カラー表示可能の機種もある)，その多くは，外部CRTディスプレイ接続のためのRGBインターフェースを持ちます．

最近の機種では256色モードを持つものがあります．PC9821/Ap/As/Ae/Ce/Af/Ne/Bp/Bs，PC98GS，PC386M等では，640×480×256色を表示可能です(PC386Mは640×400×256色)．

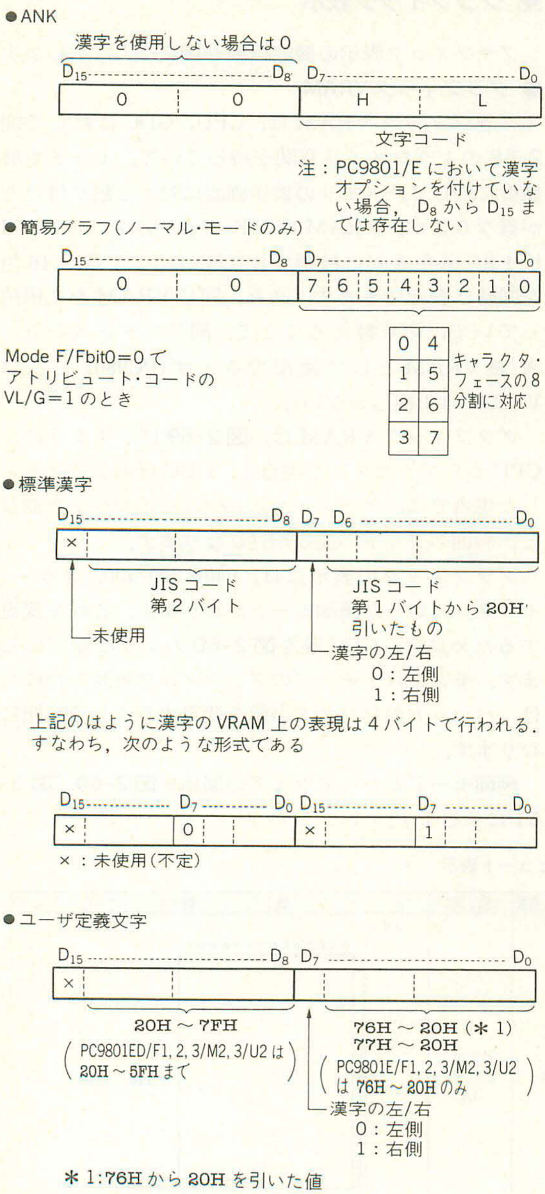
● 標準ディスプレイと専用高解像度ディスプレイ

PC98シリーズでは，主に，標準ディスプレイと専用高解像度ディスプレイで使用できます．標準ディスプレイは水平同期周波数が15.75 kHzの200ライン用ディスプレイで，400ライン表示ができないために漢字表示がサポートされず，ほとんど使われません．

専用高解像度ディスプレイは水平同期周波数が24.8 kHzの400ライン用ディスプレイで，一般的に使用されているものです．また，256色(480ライン)モードが使用可能な機種では，水平同期周波数が31.5 kHzが使用可能なディスプレイを必要とします．

ハイレゾ・モードでは，水平同期周波数が32.8 kHz，または50.0 kHz(インターレース)で，テキスト表示は80桁×25/31行，グラフィック表示は1120×750を表示できます．

〈図 2-53〉 文字コード表現



〈図 2-54〉 ANK 文字一覧

上位4ビット→ / 下位4ビット↓

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		D_E		0	@	P	\`	p				ー	タ	ミ		
1	S_H	D_1	!	1	A	Q	a	q			。	ア	チ	ム		円
2	S_X	D_2	″	2	B	R	b	r			「	イ	ツ	メ		年
3	E_X	D_3	#	3	C	S	c	s			」	ウ	テ	モ		月
4	E_T	D_4	$	4	D	T	d	t			、	エ	ト	ヤ		日
5	E_Q	N_K	%	5	E	U	e	u			・	オ	ナ	ユ		時
6	A_K	S_N	&	6	F	V	f	v			ヲ	カ	ニ	ヨ		分
7	B_L	E_B	′	7	G	W	g	w			ァ	キ	ヌ	ラ		秒
8	B_S	C_N	(	8	H	X	h	x			ィ	ク	ネ	リ	♠	
9	H_T	E_M	)	9	I	Y	i	y			ゥ	ケ	ノ	ル	♥	
A	L_F	S_B	＊	:	J	Z	j	z			ェ	コ	ハ	レ	♦	
B	H_M	E_C	+	;	K	[	k	{			ォ	サ	ヒ	ロ	♣	
C	C_L	→	,	<	L	¥	l	|			ャ	シ	フ	ワ	●	＼
D	C_R	←	−	=	M	]	m	}			ュ	ス	ヘ	ン	○	
E	S_O	↑	.	>	N	∧	n	~			ョ	セ	ホ	″		
F	S_I	↓	／	?	O	_	o				ッ	ソ	マ	°		

PC9801/E/F1, 2, 3/M2, 3/U2/VF2/VM0, 2, 4/UV2 では 60H と FCH の文字はない

◈ テキスト表示

PC98 シリーズのテキスト表示の特色は，漢字の表示は英数字同様にテキスト VRAM への書き込みだけで表示されることです．PC/AT 互換機等の DOS/V のようにグラフィック VRAM に漢字パターンを書いて表示する機種に比べて，漢字を扱う表示が格段と速くなり，比較的速度の遅い CPU を載せた機種でも，全体として満足のいく処理速度が得られることが多いようです．

テキスト表示の機能概要を図 2-51 に示します．

● テキスト VRAM

テキスト VRAM は，CPU，GDC に対して図 2-52 のようなメモリ空間を持っています．

画面に表示される文字とテキスト VRAM の書くデータの関係は，半角1文字に対して2バイト（1ワード/16 ビット）分を使用します．D_0～D_7までの下位バイトが文字コードを表し，D_8～D_{15}までの上位バイトが文字の属性，漢字コードを表します．

40 桁表示の場合は，テキスト VRAM のデータが一つおきに有効になります．例えば，1桁1行目は A0000H～A0001H の2バイトで，2桁1行目は A0004H～A0005H の2バイトになります．80 桁表示の場合は順番にすべての VRAM が有効です．

テキスト VRAM はすべて半角単位で指示します．漢字表示の場合は半角2文字分（倍の大きさ）で漢字1文字を表現しますから，右部分，左部分を別々に指定して合計4バイト（2ワード）必要になります．

漢字表示のための漢字コードは，JIS コードが基本です．具体的には，下位バイトには JIS コードの第1バイトから 20H を引いたものを，上位バイトには JIS コードの第2バイトを書きます．また，漢字の左右どちらの部分を表示するかは，下位バイト D_7のビットで決まり，$D_7=0$ で右側，$D_7=1$ で左側を表示します．ユーザ定義が可能なユーザ定義文字も漢字と同様の方式で使用します．図 2-53 に文字コード表現を，図 2-54 に ANK 文字表示一覧を示します．

● アトリビュート

テキスト VRAM のすぐ上のアドレスにアトリビュート用の RAM があり，表示される文字の色や属性を決定します．アトリビュートはテキスト VRAM と1対1で対応し，同時に変更する必要があります．

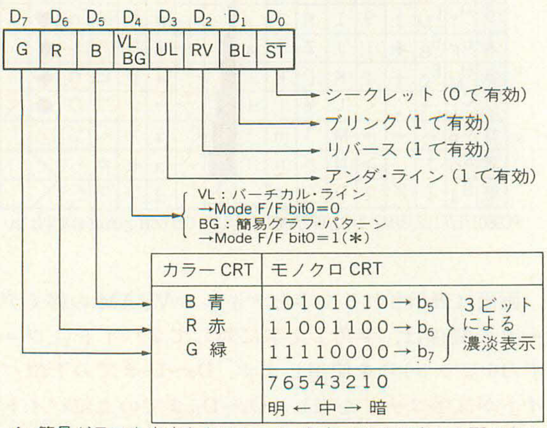
図 2-55 にアトリビュート表現，図 2-56 にアトリビュート表示を示します．

● カーソル表示

カーソルは基本的にはブリンキング・ブロック形式ですが，GDC の設定で自由な形式に設定することができます．

漢字表示のときのカーソル表示でカーソルが漢字の左側(1 バイト目)にある場合は，漢字全体にカーソル表示されますが，カーソルが漢字の右側(2 バイト目)にある場合は，漢字の右半分だけにカーソル表示されます．

〈図 2-55〉アトリビュート表現



*：簡易グラフを出すときは b_4～b_7 をすべて 0 にする必要がある
簡易グラフはノーマル・モードのみ

◈ グラフィック表示

グラフィック表示の機能概要を図 2-57 に示します．

● グラフィック VRAM

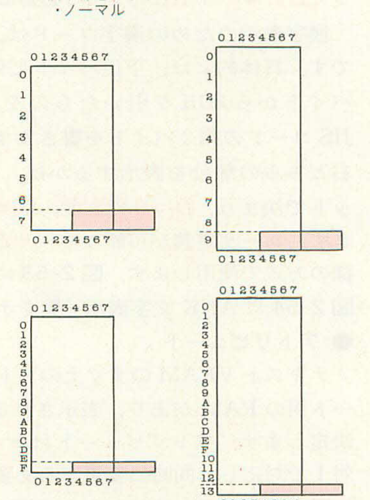
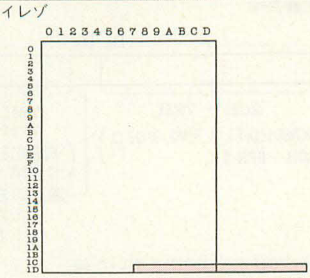
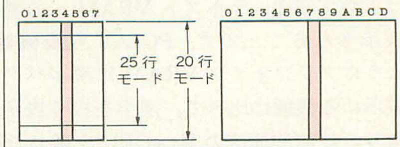
グラフィック VRAM は，CPU，GDC に対して図 2-58 のようなメモリ空間を持っていて，モードや解像度に応じて，メモリの表示画面に対する割り付け方が異なります．VRAM は 32K バイトのプレーンが 4 枚 1 組(16 色未対応機種では 3 枚)で，2^4 で最大 16 色を同時に表示できます．また，同じ VRAM を 2 組持っていて，切り替えることで，同じアドレスから，表/裏 VRAM として使用できます(PC9801/U では VRAM は 1 組しかない)．

グラフィック VRAM は，図 2-59 に示すように，CPU からアクセスした場合と，GDC 経由でアクセスした場合では，データ・バス(メモリ)のビットの並びと，画面のドットの並びが逆になります．

グラフィックの表示には，640×200/400，カラー/モノクロといった画面モードがあります．これを変更するためには，GDC 等を図 2-60 のような設定にします．モノクロ・モードのグラフィック画面の合成では，パレットのレジスタの値を変更することで可能になります．

画面モードとハードウェアの関係を図 2-60，図 2-61 に示します．

〈図 2-56〉アトリビュート表示

ビット位置	名　称	機　能
0	シークレット ST	文字を表示しない． ・UL,VL は影響を受けない． ・反転時はヌキ文字が消える．
1	ブリンク BL	点滅表示を行う． ・UL,VL は点滅しない． ・反転時はヌキ文字が点滅する．
2	リバース RV	反転表示を行う． ・UL,VL は反転しない．
3	アンダ ライン UL	横下線を表示する． ・ノーマル  UL は必ず半カラム右にずれる．ただし，80 カラム目の右半分はカットされる．色指定により，その色が出る．
3	アンダ ライン UL	・ハイレゾ  UL は必ず半カラム右にずれる．ただし，80 カラム目の右半分はカットされる．色指定により，その色が出る．
4	バーチカル ライン VL	縦線を表示する． ・ノーマル　・ハイレゾ  25 行モード　20 行モード
4	簡易グラフ パターン BG	簡易グラフ・パターンを表示する(ノーマル・モードのみ)

ビット位置	名　称	カラー CRT の色指定	モノクロ CRT の濃度指定
5	ブルー B	青	2^0
6	レッド R	赤	2^1
7	グリーン G	緑	2^2

〈図 2-57〉グラフィック表示の機能概要

	ノーマル・モード
モノクロ	640×200 ドット　16[12]画面＊4画面×4組＊合成可能 640×400 ドット　8[6]画面＊4画面×2組＊合成可能
カラー	640×200 ドット　4画面＊640×400 ドット　2画面＊ アナログ RGB ディスプレイ使用時　4096 色中 16 色(8 色)表示(16 階調濃淡表示可) ディジタル RGB ディスプレイ使用時　8 色中 8 色表示
VRAM	(32KB×4[3]プレーン)×2組＊
	GDC による描画機能あり CPU による直接 READ/WRITE 可能(GDC 描画中のアクセスは不可) グラフィック・チャージャ(GRCG，EGC)による READ/WRITE 可能(チャージャ動作時，GDC の描画は不可)

注：PC98LT は VRAM が 32KB であり，640×400 モノクロ・グラフィックのみ
　4096 色中 16 色表示は，ディップ・スイッチ SW_{1-8} ON 時(拡張グラフィック・モード)のみ可能
　＊ PC9801 および PC9801U2 では，表示可能な画面数が半分になる
　[ ]内は 16 色表示未対応の場合の数値
　　PC9801/E/F1，2，3/M2，3 では 16 色表示不可
　　PC9801U2/VF2/VM0，2，4 では 16 色グラフィック・ボードはオプション
　　PC9801/E/F1，2，3/M2，3 にはグラフィック・チャージャ機能はない
　　PC9801BA，BX，P，98NOTE，PC9821，Ap，As，Ae，Ce，Af は専用高解像度ディスプレイ固定である

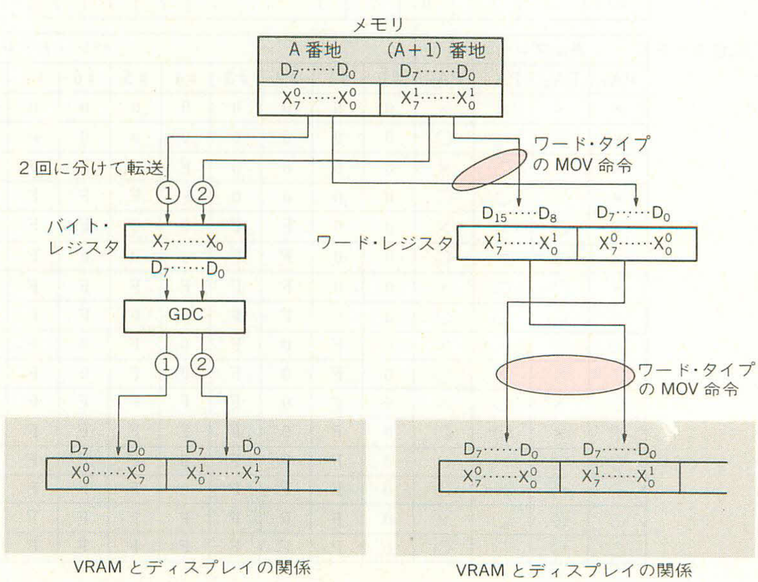
〈図 2-58〉グラフィック VRAM のメモリ空間

GDC アドレス	CPU アドレス	DATA HIGH (D_{15} … D_8) DATA LOW (D_7 … D_0)	プレーン名 200本表示 モノクロ・モード	200本表示 カラー・モード	400本表示 モノクロ・モード	400本表示 カラー・モード
4000	A : 8000	P_{00}/P_{01}(＊1) ($GVRAM_0$)	PA_{00}/PA_{01}	PA_0/PA_1	PA_{00}/PA_{01}	PA_0/PA_1
			PB_{00}/PB_{01}	PB_0/PB_1		
8000	B : 0000	P_{10}/P_{11} ($GVRAM_1$)	PA_{10}/PA_{11}	PA_0/PA_1	PA_{10}/PA_{11}	
			PB_{10}/PB_{11}	PB_0/PB_1		
C000	B : 8000	P_{20}/P_{21} ($GVRAM_2$)	PA_{20}/PA_{21}	PA_0/PA_1	PA_{20}/PA_{21}	
			PB_{20}/PB_{21}	PB_0/PB_1		
(＊2) 00000	E : 0000	P_{30}/P_{31} ($GVRAM_3$)	PA_{30}/PA_{31}	PA_0/PA_1	PA_{30}/PA_{31}	PA_0/PA_1
			PB_{30}/PB_{31}	PB_0/PB_1		

→バイト・アドレス(アドレスの最下位により H，L バイトを切り分ける)
→ワード・アドレス(H，L バイトは同時にアクセス)

＊1：上記空間において，PC9801/U2 では P_{01}，P_{11}，P_{21}，P_{31} は存在しない
＊2：8 色モードのときはバッファが Enable にならない
　　I/O ポート(42H)bit3＝1 のときは 16 色表示対応

〈図 2-59〉
グラフィック VRAM へのアクセスの方法



〈図 2-60〉画面モードとハードウェアの関係

表示状態 CRT	表示状態 グラフィック・モード	表示状態 グラフ解像度	表示状態 表示プレーン (＊2)	設定値 GDC L/F	設定値 GDC L/R	設定値 GDC SAD	設定値 Palette Reg	設定値 Mode F/F bit 1	設定値 Mode F/F bit 4
専用高解像度ディスプレイ	カラー	640×200	PA_1	400	2	0	各コードのRGB	0	(＊1) 1
			PB_1			1F40H			
		640×400	PA_1		1	0			0
	モノクロ	640×200	PA_{01} PA_{11} PA_{21} (PA_{31})		2	0	画面合成コード	1	(＊1) 1
			PB_{01} PB_{11} PB_{21} (PB_{31})			1F40H			
		640×400	PA_{01} PA_{11} PA_{21} (PA_{31})		1	0			0
高解像度および標準ディスプレイ	カラー	640×200	PA_1	200	1	0	各コードのRGB	0	0
			PB_1			1F40H			
	モノクロ	640×200	PA_{01} PA_{11} PA_{21} (PA_{31})			0	画面合成コード	1	
			PB_{01} PB_{11} PB_{21} (PB_{31})			1F40H			

(　)内は，16色表示未対応時は無効

＊1：1を設定した場合は画面が1本おきの表示になる
0を設定した場合は画面に2本同じ走査線が表示される

＊2：PA_1，PA_1 の i=1 のとき，I/O ポート・アドレス 0A4H に 01H を OUT する
i=0 のとき，I/O ポート・アドレス 0A4H に 00H を OUT する
PC9801/U2 の場合，表示プレーンは i=0 プレーンのみ使用可能

＊3：PC9801BA，BX，P，98NOTE，PC9821，Ap，As，Ae，Ce，Af は専用高解像度ディスプレイ固定である

〈図 2-61〉
画面合成コード(モノクロ・モード)

8 色モード

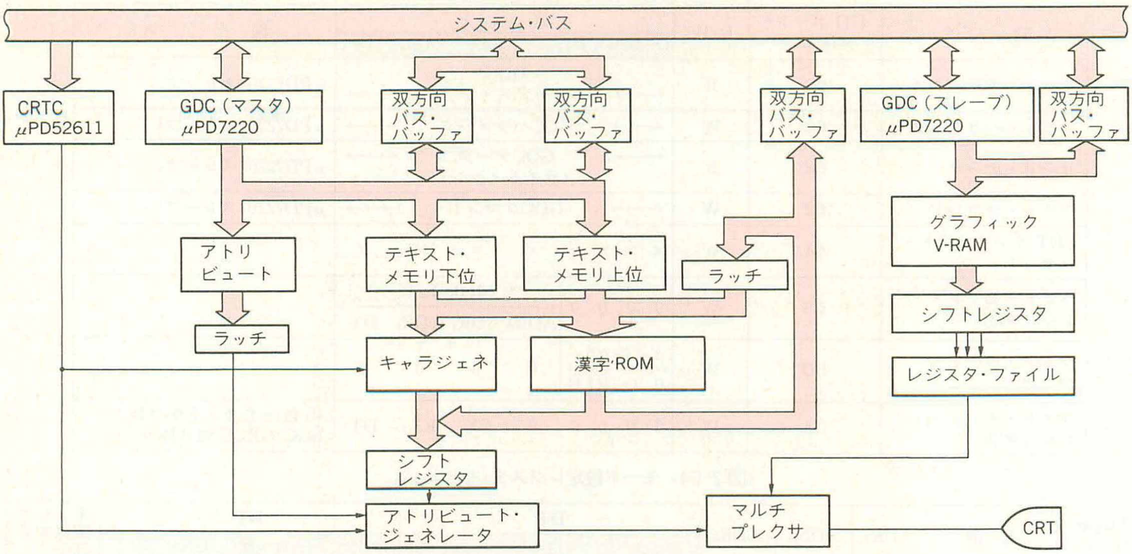
表示プレーン PA_{01}	PA_{11}	PA_{21}	パレット・レジスタの値 #0	#1	#2	#3	#4	#5	#6	#7
×	×	×	0	0	0	0	0	0	0	0
×	×	○	0	0	0	0	7	7	7	7
×	○	×	0	0	7	7	0	0	7	7
○	×	×	0	7	0	7	0	7	0	7
×	○	○	0	0	7	7	7	7	7	7
○	×	○	0	7	0	7	7	7	7	7
○	○	×	0	7	7	7	0	7	7	7
○	○	○	0	7	7	7	7	7	7	7

＊　×印は画面合成に関係しないプレーン

16 色モード

表示プレーン PA_{01}	PA_{11}	PA_{21}	PA_{31}	パレット・レジスタの値 #0	#1	#2	#3	#4	#5	#6	#7	#8	#9	#A	#B	#C	#D	#E	#F
×	×	×	×	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
×	×	×	○	0	0	0	0	0	0	0	0	F	F	F	F	F	F	F	F
×	×	○	×	0	0	0	0	F	F	F	F	0	0	0	0	F	F	F	F
×	×	○	○	0	0	0	0	F	F	F	F	F	F	F	F	F	F	F	F
×	○	×	×	0	0	F	F	0	0	F	F	0	0	F	F	0	0	F	F
×	○	×	○	0	0	F	F	0	0	F	F	F	F	F	F	F	F	F	F
×	○	○	×	0	0	F	F	F	F	F	F	0	0	F	F	F	F	F	F
×	○	○	○	0	0	F	F	F	F	F	F	F	F	F	F	F	F	F	F
○	×	×	×	0	F	0	F	0	F	0	F	0	F	0	F	0	F	0	F
○	×	×	○	0	F	0	F	0	F	0	F	F	F	F	F	F	F	F	F
○	×	○	×	0	F	0	F	F	F	F	F	0	F	0	F	F	F	F	F
○	×	○	○	0	F	0	F	F	F	F	F	F	F	F	F	F	F	F	F
○	○	×	×	0	F	F	F	0	F	F	F	0	F	F	F	0	F	F	F
○	○	×	○	0	F	F	F	0	F	F	F	F	F	F	F	F	F	F	F
○	○	○	×	0	F	F	F	F	F	F	F	0	F	F	F	F	F	F	F
○	○	○	○	0	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F

〈図 2-62〉CRT コントロール・ブロック



GDC と周辺 LSI

▶使用 LSI
μPD7220 相当×2，μPD52611
▶ I/O アドレス
60H，62H（GDC/マスタ）
A0H，A2H（GDC/スレーブ）
70H，72H，74H，76H，78H，7AH（CRTC）

◈ GDC と周辺 LSI のハードウェア

PC98 シリーズでは，CRT インターフェースに GDC（μPD7220）が 2 個使用されています．そのうち 1 個はテキスト画面用でマスタ動作で，もう一つはグラフィック画面用でスレーブ動作になっています．マスタの GDC は，60H，62H，スレーブの GDC は A0H，A2H で操作できます．他にもタイミング関係用に CRTC（μPD52611）等が使用されています．

垂直同期信号を出しているマスタの GDC は，テキスト VRAM のアドレス生成，CRT の同期信号の生成を行い，CRT 関係回路の核になっています．スレーブの GDC はマスタからの同期信号に合わせて，グラフィック画面を制御します．

VRAM からの出力信号は，テキストならばキャラクタ・ジェネレータ（CG・漢字 ROM 等）を通り，グラフィックならばそのままシフトレジスタでパラレル-シリアル変換され，必要に応じてパレット操作や，アナログ RGB のための D-A コンバータを経てビデオ信号になります．

CRTC は，主にテキスト用の垂直同期関係の制御を行い，CG ライン・カウンタの出力，アドレス加算回路へのタイミング出力，アンダライン等のタイミング出力を行っています．それらの制御用に「ライン・カウンタ制御命令」として，70H〜7AH までの 6 個のレジスタを持ちます．

CRT コントロールのブロック図を図 2-62 に示します．

◈ テキスト表示制御用命令

テキスト表示関連の I/O アドレスとしては以下のようなものがあります．

60H，62H	GDC（マスタ）
64H	CRT インタラプト・リセット
68H	モード・レジスタ 1 設定
6CH	ボーダ・カラー設定
6AH	モード・レジスタ 2 設定

CRT インタラプト・リセット 64H は，スムース・スクロール制御のための割り込みリセットです．CRTV（VSYNC）割り込みを発生するためのもので，64H にライト（リセット）すると，一度だけ CRT の VSYNC 信号の立ち上がり時に「ベクタ番号 0AH」の割り込みがかかります．通常は連続して割り込みを使用するので，CRTV 割り込みプログラムの中で，このポートをアクセス（リセット）します．この場合，約 16 ms 間隔で割り込みが発生します．

CRTV 割り込みは，電源投入後に 1 回だけ割り込むだけで，リセットしない限り，その後は割り込みません．

モード・レジスタ 1 設定 68H は，主に画面表示に関連するモードの変更をします．設定は $ADR_{0\sim2}$ に変更したい項目，DT には変更したいモードを入れて書き込みます．

▶ ATR SEL

〈図 2-63〉テキスト表示制御命令

命令	I/O ポート・アドレス	R/W	データ D_7 D_6 D_5 D_4	D_3	D_2	D_1	D_0	備考
リード・ステータス	60	R	←― GDC ステータス・フラグ ―→					μPD7220(スレーブ)
ライト・パラメータ	60	W	←― GDCパラメータ ―→					μPD7220(スレーブ)
リード・データ	62	R	←― GDCデータ (ライト・ペン) ―→					μPD7220(スレーブ)
ライト・コマンド	62	W	←― GDCコマンド ―→					μPD7220(スレーブ)
CRT インタラプト・リセット	64	W	× × × ×	×	×	×	×	
ライト・モード・レジスタ(1)	68	W	0 0 0 0	Mode F/F ADR₂	ADR₁	ADR₀	DT	
ライト・ボーダ・カラー	6C	W	ボーダ・カラー 0 G R B	0	0	0	0	
ライト・モード・レジスタ(2)	6A	W	0 0 0 0	0	EX_1	EX_2	DT	16 色/8 色モード切り替え EGC/GRCG 切り替え

〈図 2-64〉モード設定レジスタ(モード 1)

Mode F/F	名前	ADR_2	ADM_1	ADR_0	DT 1	DT 0	主として関係する部分
0	ATR SEL	0	0	0	ATR7 が簡易グラフ	ATR7 が バーティカル・ライン	テキスト
1	GRAPHIC Mode	0	0	1	モノクロ・グラフィック・モード	カラー・グラフィック・モード	グラフ
2	Column WIDTH	0	1	0	40 字モード	80 字モード	テキスト
3	FONT SEL	0	1	1	文字フォントの大きさ 7×13	6×8	テキスト
4	GRP Mode	1	0	0	専用高解像度ディスプレイを 200 本モード・グラフで使用する	・専用高解像度 400 本 ・標準解像度	グラフ
5	KAC Mode	1	0	1	漢字アクセス・モード ビット・マップ	コード・アクセス	漢字
6	NVMW PERMIT	1	1	0	不揮発メモリへの書き込み PERMIT	INHIBIT	
7	DISP ENABLE	1	1	1	表示可とする	すべての画面を表示しない	テキスト

テキスト VRAM のアトリビュートの D_4で設定する簡易グラフと，バーチカル・ラインの切り替えをします．

▶ GRAPHIC Mode

モノクロとカラーの切り替えをします．

▶ Column WIDTH

40 桁/80 桁の切り替え，40 桁時は一つおき表示されます．

▶ FONT SEL

表示されるフォントを切り替えます．6×8 モードでは漢字の表示ができません．

▶ GRP Mode

専用高解像度ディスプレイで 200 ライン表示モードを使用するときに「1」にします．400 ライン・モード時に使用すると，1 行おきに表示されます．

▶ KAC Mode

漢字アクセス・モードを変更します．CG ウィンドウやキャラクタ・ジェネレータ制御命令で使用します．

▶ NVMW PERMIT

不発揮メモリへの書き込みを許可/禁止を指定します(メモリ・スイッチ)．

▶ DISP ENABLE

画面の表示の許可/禁止を指定します．

「ボーダ・カラー設定(6CH)」は，描画画面の外側のカラーを変更します．専用高解像度ディスプレイでは，描画画面の上下の色が出ません．

「モード・レジスタ 2 設定(6AH)」は，16 色/8 色モードの切り替えをします．

EGC の拡張モード指定にも使用されます．

テキスト表示制御用命令を図 2-63 に，モード設定レジスタ 1 を図 2-64 に，モード設定レジスタ 2 を図 2-65 に示します．

〈図 2-65〉モード設定レジスタ(モード 2)

ADR 6 5 4 3 2 1 0	名前	DT 1	DT 0
0 0 0 0 0 0 0	COLOR SEL	16 色モード	8 色モード

ハードウェア・リセット時 0 側(8 色モード)になる

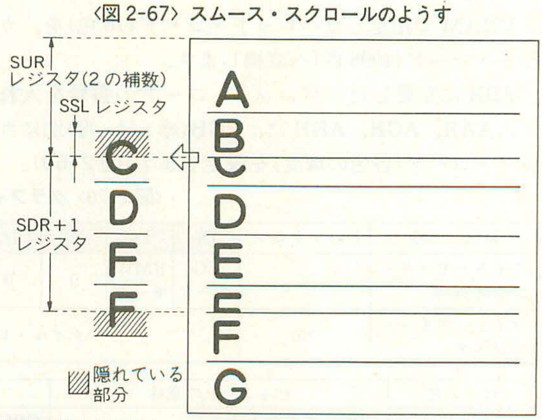
〈図 2-66〉ライン・カウンタ制御命令

命令	I/Oポート・アドレス	R/W	データ $D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0$	備考
ライトPL	70	W	キャラクタ位置ライン数 (PL)(＊1)	ライン・カウンタの初期値 (ボディーフェースのうち，キャラクタが表示される位置の上から数えたライン数の2の補数)
ライトBL	72	W	ボディーフェース・ライン数 (BL)(＊1)	キャラクタの先頭を0としたときのボディーフェース下端のライン数
ライトCL	74	W	キャラクタ・ライン数 (CL)(＊1)	キャラクタ・フェースのライン数
ライトSSL	76	W	スムース・スクロール・ライン数 (SSL)	スクロール・エリア内の文字がスクロールしているライン数
ライトSUR	78	W	スクロール・エリア上辺位置行数 (SUR)	スクロール・エリアの上辺の位置の行数の2の補数 (この次の行よりスクロールする)
ライトSDR	7A	W	スクロール・エリア行数 (SDR)	(スクロール・エリアの行数)－1

		25行	20行
PL	専用高解像度ディスプレイ	00H	1EH
	高解像度および標準ディスプレイ	00H	1FH
BL	専用高解像度ディスプレイ	0FH	11H
	高解像度および標準ディスプレイ	07H	08H
CL	専用高解像度ディスプレイ	10H	
	高解像度および標準ディスプレイ	08H	

＊1：初期設定値

〈図 2-67〉スムース・スクロールのようす



※画面最上位行からスムース・スクロールさせる場合は SUR＝1FH，2行目からならSUR＝1EHになる

◈ ライン・カウンタ制御用命令

ライン・カウンタ制御回路はCGのライン・カウンタ出力やアンダラインのタイミング出力，スムース・スクロール機能を実現するためのアドレス加算回路へタイミング出力等，CRTの垂直方向の制御信号を出力します(図2-66)．

◈ スムース・スクロール制御用命令

スムース・スクロールは主にライン・カウンタ制御回路によって実現されています．スムース・スクロールさせるためには，スクロール・エリアの上辺行数(SUR)，スクロールする行数(SDR)をセットして，画面描画のブランキングのタイミング(CTRV/SYNC割り込み)でスムース・スクロール・ライン数(SSL)を増減させることで，ちらつきがなく，画面がスムースにアップ・ダウンできます．

スムース・スクロール・エリア(SUR，SDRで設定)では，SSLだけ上にずれて(加算されて)表示されます．スクロール・エリアの下辺行では，その1行下の文字の上の部分が表示されてしまいますので，下辺行とその下の行の文字がだぶって表示されることになります．これを防ぐためには，スクロール・エリアより下の部分を，GDCのSCROLLコマンド等で画面を分割して，スクロール・エリアと分離させておきます．図2-67がスムース・スクロールのようすです．

◈ グラフィック表示制御用命令

テキスト表示関連のI/Oアドレスとしては以下のようなものがあります．

A0H，A2H　GDC(スレーブ)
A4H　表示画面選択レジスタ
A6H　描画画面選択レジスタ
A8H　パレット・レジスタ(パレット番号)
AAH　パレット・レジスタ(GREEN)
ACH　パレット・レジスタ(RED)
AEH　パレット・レジスタ(BLUE)

表示画面選択レジスタ(A4H)は，2画面(裏/表)あるVRAMのうち，表示する画面を指定します．00Hで表VRAM(P_{00}，P_{10}，P_{20}，P_{30})，01Hで裏VRAM(P_{01}，P_{11}，P_{21}，P_{31})を選択します．

描画画面選択レジスタ(A6H)は，2画面(裏/表)あるVRAMのうち，描画(CPUの読み書き)する画面を指定します．00Hで表VRAM(P_{00}，P_{10}，P_{20}，P_{30})，01Hで裏VRAM(P_{01}，P_{11}，P_{21}，P_{31})を選択します．表示画面と描画画面を別々に設定できるので，描画中に，表示画面のちらつきをなくせます．

パレット・レジスタ(A8H〜AEH)は，8色モードと16色モードで設定方法が違います．

▶ 8色モード時

VRAMで指定したパレット・コード(8色)を，別のカラー・コード(8色)へ変換します．

一つのパレット・コードにつき，一つのパレット・レジスタを持ち，カラー・コードを3ビットで表します．

〈図 2-68〉 パレット・レジスタ 8 色モード

パレット・コード	アドレス	カラー・コード(R=赤, G=緑, B=青)							
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	AEH	0	B	G	R	×	×	×	×
1	AAH	0	B	G	R	×	×	×	×
2	ACH	0	B	G	R	×	×	×	×
3	A8H	0	B	G	R	×	×	×	×
4	AEH	×	×	×	×	0	B	G	R
5	AAH	×	×	×	×	0	B	G	R
6	ACH	×	×	×	×	0	B	G	R
7	A8H	×	×	×	×	0	B	G	R

A8H～AEH までの 4 バイトのレジスタを，上位 4 ビット，下位 4 ビットに分けて使用して，合計 8 個分のパレット・レジスタとして使います．4 ビット中上位 1 ビットは未使用です(図 2-68)．

▶ 16 色モード時

VRAM で指定したパレット・コード(16 色)を，カラー・コード(4096 色)へ変換します．

A8H に変更したいパレット・コードの番号を入れて，AAH，ACH，AEH に，RGB(赤・緑・青)別にカラー・コード(各色の輝度)を設定します(図 2-69)．

〈図 2-69〉 パレット・レジスタ 16 色モード

アドレス	内　容	データ
A8H	パレット番号	00H-0FH
AAH	GREEN の輝度	00H-0FH
ACH	RED の輝度	00H-0FH
AEH	BLUE の輝度	00H-0FH

◆ グラフィック・チャージャ(GRCG)制御命令

GRCG は，CPU と VRAM の間に専用のハードウェアを入れることで，一度の読み書きで，4 枚ある VRAM すべてにアクセスできるので，グラフィック描画を高速に行えます．特に画面を同じ色で塗る場合には効果を発揮します．GRCG は，PC9801/E/F/M では使用できません．

GRCG の制御レジスタは二つあり，これで VRAM アクセス時のモード(効果)を変更します．

GRCG 制御命令を図 2-70 に示します．

▶ TDW モード

CPU が VRAM に書き込むと CPU のデータは無視され，タイル・レジスタの内容(8 ビット・レジスタが

〈図 2-70〉 グラフィック・チャージャ制御命令

命　令	I/O アドレス	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
ライト・モード・レジスタ	7C	CG モード	RMW モード	0	0	$\overline{P_3EN}$	$\overline{P_2EN}$	$\overline{P_1EN}$	$\overline{P_0EN}$
ライト・タイル・レジスタ	7E	タイル・レジスタ 0～3 (*)							

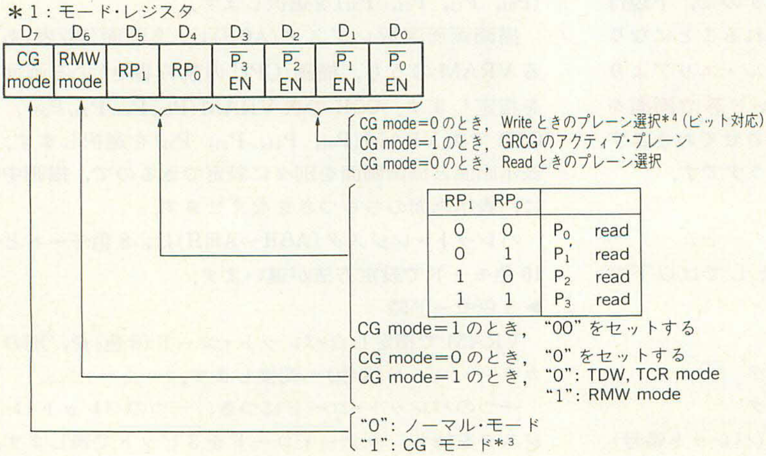
＊：モード・レジスタにライトを行うと，タイル・レジスタ 0 ライトにリセットされる．その後ライトするたびに，書かれるレジスタはレジスタ 1，レジスタ 2，レジスタ 3，レジスタ 0，と変化する

ビット名	ビット=1 の意味	ビット=0 の意味
CG モード	GRCG を有効とする CPU の VRAM アクセスをきっかけとして，GRCG が各モードの動作を実行する	GRCG を無効とする CPU の VRAM アクセスは，そのまま VRAM のリード(ライト)となる
RMW モード	CPU の VRAM ライトにより，RMW モードの動作を行う CPU の VRAM リードは無視される	CPU の VRAM ライトにより TDW モードの動作を行う CPU の VRAM リードにより TCR モードの動作を行う
P_3EN，P_2EN P_1EN，P_0EN	該当するプレーンを無効とする	該当するプレーンを有効とする 複数ビットの指定が可能 GRCG は，有効となっているプレーンに対してのみアクセスを行う

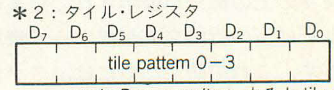
＊ GRCG 動作中は，CPU に WAIT がかかり，モード・レジスタの変更が不可となる

＊ GRCG 動作中は，バスが占有されるため DMA 転送レートが低下する．原則として DMA と GRCG は同時に使用しないこと(例えば，固定ディスクの DMA 転送などと，GRCG に対するストリング命令によるアクセスは同時に行わないようにする必要がある)

＊1：モード・レジスタ



＊2：タイル・レジスタ

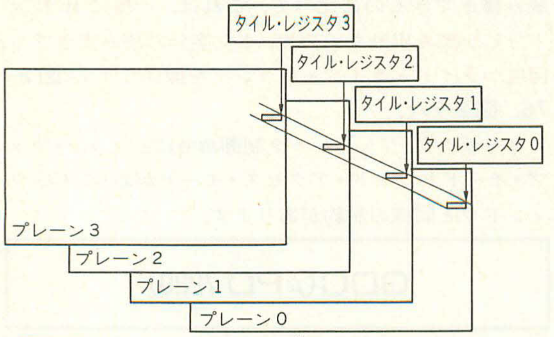


mode Reg に write にすると tile Reg 0 write にリセットされ，その後書き込むたびに tile Reg 1, tile Reg 2, tile Reg 3, tile Reg 0… となる．

＊3：グラフィック・チャージャ動作中 (CG mode) は GDC からの描画は禁止される．GDC に描画命令を出しても，VRAM への直接アクセスは起こらない．モード・レジスタはグラフィック・チャージャ動作中は変更できない

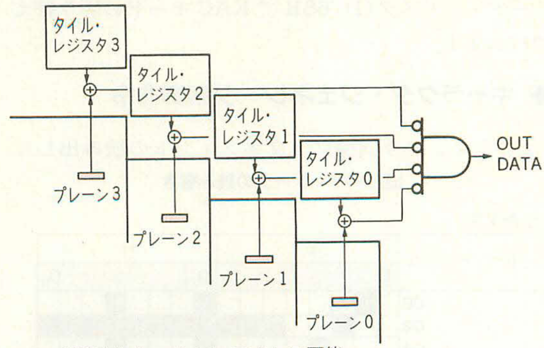
＊4：この設定は CGmode=0 のときは GDC の描画に対しても有効 (READ プレーンと WRITE プレーンを変えて設定すると，プレーン間の転送が可)

〈図 2-71〉TDW モード



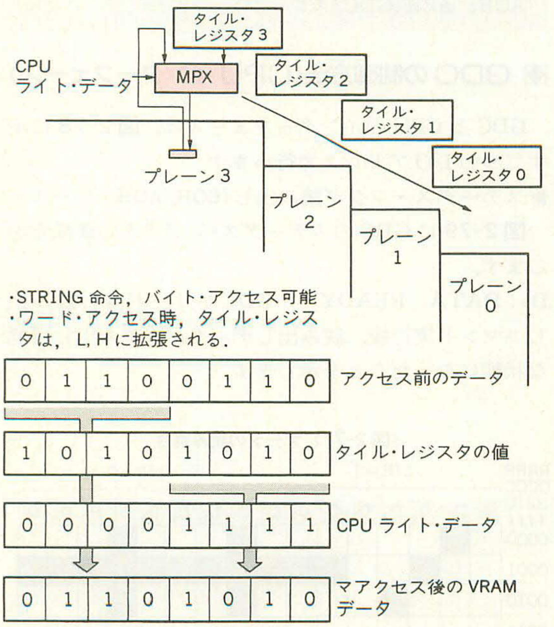
・STRING 命令，バイト・アクセス可能
・ワード・アクセス時，タイル・レジスタ・パターンはL,Hに拡張される．

〈図 2-72〉TCR モード



・SCAN 命令，バイト・アクセス可能．
・ワード・アクセス時，タイル・レジスタ・パターンは，H, L に拡張される．
・色の検出ではタイル・レジスタをすべて1かすべて0にセットする．
・アクティブにしないプレーンは無視される．

〈図 2-73〉RMW モード



・STRING 命令，バイト・アクセス可能
・ワード・アクセス時，タイル・レジスタは，L, H に拡張される．

4 個)を VRAM に書き込みます．画面を同じ色で塗る場合には効果があります(図 2-71)．

▶ TCR モード

CPU が VRAM を読み出すと，各プレーンとタイル・レジスタの内容を比較し，一致したビットを「1」として CPU に読み込みます．画面の色を識別するのに効果があります(図 2-72)．

▶ RMW モード

CPU が VRAM に書き込むと，CPU のデータのうち，ビットが 1 である部分は，タイル・レジスタの内容が VRAM に書かれ，ビットが 0 である部分は，元のデータのままになります．TDW モードをビット単位で行うことができますので，ドットやラインを引く場合には有効です(図 2-73)．

リスト 2-5 に GRCG テストの参考プログラムを示します．

〈リスト 2-5〉グラフィック・チャージャのテスト・プログラム

```
/*
**  GRCGテスト
**
**  VRAM消去
**
*/
#include <dos.h>

int main()
{
    int             i;
    unsigned int    adres;

    outportb(0x7c,0x80);                /*  TDWモード、全プレーン有効   */
    for (i = 0; i < 4; i++){
        outportb(0x7e,0);               /*  タイルレジスタ設定          */
    }

    for (adres = 0; adres < 32*1024l; adres+=2){
        poke(0xa800,adres,0);
    }
    outportb(0x7c,0);                   /*  GRCG無効                    */

    return 0;
}
```

〈図 2-74〉CG ウィンドウ・アドレス

命　令	I/O アドレス	R/W
Write 2nd byte code	0A1H	W
Write 1st byte code	0A3H	W

◈ CG ウィンドウ

CG ウィンドウは，メモリ空間の一部(A4000H～A4FFFH)に，文字フォントのデータを出して，読み書きができます．フォント・データが CG ウィンドウのメモリ空間より大きいために，フォント・データの一部分をウィンドウ(窓)から見る形で使います．

PC9801VM 以前の機種や，ラップトップ，ノート等では，CG ウィンドウを搭載していない機種もあります．CG ウィンドウ・アドレスを図 2-74 に，データの読み書きを図 2-75 に示します．

CG ウィンドウで漢字フォントを読み書きする場合には，ビット・マップ・モードとコード・アクセス・モードがあります．コード・アクセス・モードは高速に読み書きできますが，GDC が V-SYNC 中でないと画

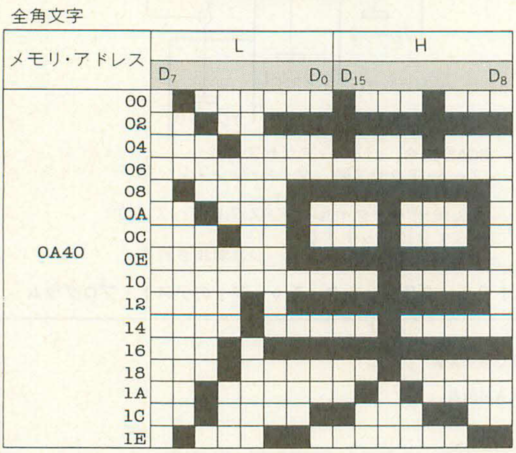
面が乱れることがあります．この切り替えはライト・モード・レジスタ(1)/68H で KAC モードの切り替えで行います．

◆ キャラクタ・ジェネレータ制御命令

CG ウィンドウ同様に，文字フォントの読み出しに使用します．CG ウィンドウがフォント全体を一度に読み書きできるのに比べて，これは，一度に 16 ビット分しか読み出せないので，1 文字分の読み書きでも，何度かライト・ライン・カウンタを操作します(図 2-76，図 2-77)．

キャラクタ・ジェネレータ制御命令にもビット・マップ・モードとコード・アクセス・モードがあり，CG ウィンドウと同様の制約があります．

〈図 2-75〉データの読み書き



第 1 バイトが以下のものを除く
・2C～2F：全角ケイ線特殊文字
・76：ユーザ定義
・79～7C：拡張漢字

〈図2-76〉キャラクタ・ジェネレータ制御命令

命　　令	I/O ポート・アドレス	R/W	データ D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
ライト文字コード第 2 バイト	A1	W	←── 文字コード 第 2 バイト ──→
ライト文字コード第 1 バイト	A3	W	←── 文字コード 第 1 バイト ──→
ライト・ライン・カンウタ(＊1)	A5	W	0 0 L/R RC4 RC3 RC2 RC1 RC0
リード文字パターン〈図2-77参照〉	A9	R	CG リード・パターン ←── 左　右 ──→
ライト文字パターン〈図2-77参照〉	A9	W	CG ライト・パターン ←── 左　右 ──→

＊1：L/R は全角漢字左(1)，右(0)を指定する．
RC_0～RC_4はキャラクタ・パターンの上から何ラインかを 16 進数で指定する．ただし，CG のパターンは 16 ライン固定なので RC_4は 0 としておく(未定義でよい)

〈図 2-78〉GDC と CPU のインターフェース

アドレス	モード	機　　能
60H/A0H	リード	GDC ステータス・フラグの読み出し
60H/A0H	ライト	パラメータの書き込み
62H/A2H	リード	データの読み出し
62H/A2H	ライト	コマンドの書き込み

GDC(μPD 7220)

▶ I/O アドレス
60H，62H(GDC/マスタ)
A0H，A2H(GDC/スレーブ)

◆ GDC の制御方法(CPU インターフェース)

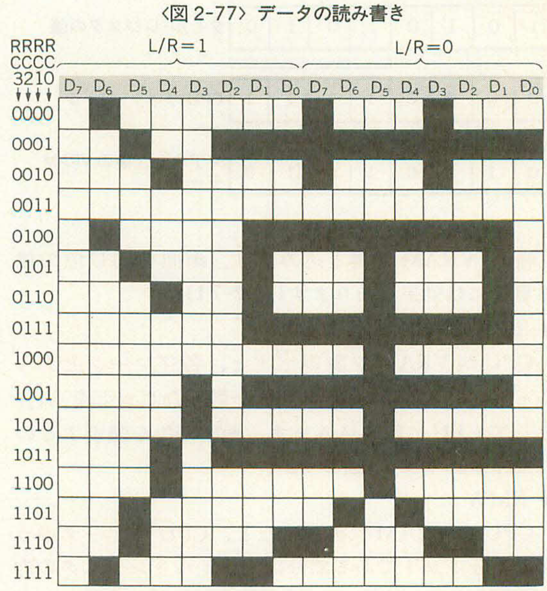
GDC と CPU のインターフェースは，図 2-78 に示す二つの I/O アドレスで行います．

▶ステータス・フラグ読み出し(60H/A0H・リード)

図 2-79 に GDC のステータス・レジスタの構成を示します．

D_0：DATA　READY……GDC がリード等の読み出しコマンド実行後，読み出しデータが，読み出し可能な状態になったことを示します．

〈図 2-77〉データの読み書き



〈図 2-79〉
GDC ステータス・レジスタ

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
LIGHT PEN DETECT	HORIZONTAL SYNC	VERTICAL SYNC	DMA EXECUTIVE	DRAWING	FIFO EMPTY	FIFO FULL	DATA READY

〈図 2-80 (a)〉 GDC のコマンド/パラメータ(動作制御)

コマンド名	機能	C/P	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	パラメータ解説
			コマンドまたはパラメータ・コード								
RESET	初期化	C	0	0	0	0	0	0	0	0	
SYNC	動作モード，同期タイミングの定義	C	0	0	0	0	1	1	1	DE	DE=0:表示停止，DE=1:表示開始
		P_1	0	0	CHR	F	I	D	G	S	CHR=1:文字モード，F=1:フラッシュレス描画 I=1:インターレース・モード，D=1:ダイナミックRAM G=1:グラフィック・モード，S=1:インターレース・シュリンク・モード(I=1)
		P_2	← C/R →								C/R:1行の表示文字数
		P_3	← VS_L →			← HS →					HS:水平同期期間
		P_4	← HFP →						← VS_H →		VS:垂直同期期間
		P_5			← HBP →						HFP:水平右側非表示期間 HBP:水平左側非表示期間
		P_6			← VFP →						VFP:垂直上側非表示期間
		P_7	← L/F_L →								VBP:垂直下側非表示期間
		P_8	← VBP →						← L/F_H →		L/F:1画面の表示ライン数
MASTER/SLAVE	マスタ動作，スレーブ動作の選択	C	0	1	1	0	1	1	1	M	M=1:マスタ動作，M=0:スレーブ動作

〈図 2-80 (b)〉 GDC のコマンド/パラメータ(映像メモリ制御)

コマンド名	機能	C/P	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	パラメータ解説
			コマンドまたはパラメータ・コード								
WRITE	ドット修正モード設定 データの書き込み	C	0	0	1	WLH		0	MOD		WLH=00:ワード転送　MOD=00:REPLACE =01:未定　=01:COMPLEMENT
	繰り返し送出可 バイト転送時は$CODE_L$のみでよい	P_1	← $CODE_L$ →								=10:下位バイト転送　=10:CLEAR =11:上位バイト転送　=11:SET
		P_2	← $CODE_H$ →								CODE:V-RAM書き込みデータ
READ	映像メモリの読み出し	C	1	0	1	WLH		0	MOD		WRITEコマンドに同じ
DMAW	映像メモリへのDMA転送指示	C	0	0	1	WLH		1	MOD		WRITEコマンドに同じ
DMAR	映像メモリからのDMA転送指示	C	1	0	1	WLH		1	MOD		WRITEコマンドに同じ

D_1：FIFO FULL……FIFOがデータでいっぱいになったことを示します．

D_2：FIFO EMPTY……FIFOが空であることを示します．

D_3：DRAWING……GDCが描画中であることを示します．

グラフィックの描画時には描画開始から終了まで“1”になっていますが，テキスト描画時には内蔵RAMからGDC内部のレジスタに，その内容が転送されるたびに“0”になります．また，DMAR/DMAR/READ/WRITEコマンド実行中はGDCが描画サイクルにあるときだけ“1”になります．

D_4：DMA EXECUTE……DMA転送を続行中であることを示します．PC98シリーズではGDCとDMAが接続されていないために，DMAは使用できません．

D_5：VERTICAL SYNC……垂直同期信号が発生していることを示します．

D_6：HORIZONAL BLANK……水平同期信号が発生していることを示します．

D_7：LIGHT PEN DETECT……ライト・ペン信号によるアドレスの検出があったことを示します．

▶パラメータ書き込み(60H/A0H・ライト)とコマンド書き込み(62H/A2H・ライト)

FIFO FULL=0であることを確認したうえで書き込みます．または，FIFO EMPTY=1であることを事前に確認をして，16バイトまでのコマンドを一気に送ることもできます．17バイト目以降はFIFO FULLを確認しながら送ります．

▶データ読み出し(62H/リード)

DATA READY=1であることを確認したうえで読み出します．FIFOは，RESETコマンド直後はCPU→GDCの方向になっています．しかし，READ/CSRR/LPEN等のコマンドが実行されたときGDC→CPUの方向へ変化します．したがって，これらのコマンドの直後は，CPU→GDC方向のFIFOは動きません．この3種類以外のコマンドを実行すると，CPU→GDCの方向へ戻ります．

◈ GDCのコマンド(動作制御)

図2-80にGDCコマンド/パラメータ一覧を示します．

▶ RESET

初期化コマンドです．基本タイミング回路，FIFOのクリア，描画動作停止，画面表示停止等です．

〈図 2-80 (c)〉GDC のコマンド/パラメータ(表示制御)

コマンド名	機能	C/P	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	パラメータ解説
			コマンドまたはパラメータ・コード								
START	表示開始	C	0	1	1	0	1	0	1	1	} どちらでもよい
			0	0	0	0	1	1	0	1	
STOP	表示停止	C	0	0	0	0	1	1	0	0	
ZOOM	拡大係数の設定	C	0	1	0	0	0	1	1	0	ZR:拡大表示時の拡大係数 ZW:拡大描画時の拡大係数
		P	← ZR →				← ZW →				
SCROLL	表示開始アドレス, 表示領域の設定	C	0	1	1	1	← RA →				RA:データ RAM のアドレス
	画面をn個に分割した場合, このパラメータをn回送出. 文字モード:$n$max=4 文字/グラフィック・モードと$n$max=2グラフィック・モード	P_{n1}	← SAD_{nL} →								SAD_n:n番目の区間の開始アドレス SL_n:n番目の区間の長さ *:表示アドレス・インクリメント量 IM:文字/グラフィック・モード時 IM=0…文字表示領域 IM=1…グラフィック表示領域 (その他のモード IM=0)
		P_{n2}	0	0	0	← SAD_{nH} →					
		P_{n3}	← SL_nL →				0	0	0	0	
		P_{n4}	*	IM	← SL_{nH} →						
CSRFORM	カーソル形状などの指定	C	0	1	0	0	1	0	1	1	L/R:1行中の表示ライン数(グラフィック・モード時…1) CS=1:カーソル表示あり BD=1:ブリンキングなし(CS=1なら常時点灯) CST:カーソル表示開始ライン値 BL:カーソル点滅周期 CFI:カーソル表示終了ライン値
		P_1	CS	0	0	← L/R →					
		P_2	← BL_L →		BD	← CST →					
		P_3	← CFI →					← BL_H →			
PITCH	映像メモリの水平方向ワード数の設定	C	0	1	0	0	0	1	1	1	水平方向のワード(16ビット)数
		P	← P →								
LPEN	ライトペン・アドレスの検出	C	1	1	0	0	0	0	0	0	LAD:ライトペン・アドレス
	読み出し専用	P_1	← LAD_L →								
		P_2	← LAD_M →								
		P_3							← LAD_H →		
VECTW	描画に必要なパラメータタの設定	C	0	1	0	0	1	1	0	0	L=1:直線描画 T=1:傾斜しないグラフィックス文字描画 C=1:円および円弧の描画 R=1:四辺形描画 SL=1:傾斜したグラフィックス文字(T=1) DGD=1:グラフィック描画(文字/グラフィック・モード)
		P_1	SL	R	C	T	L	← DIR →			
		P_2	← DC_L →								
		P_3	0	DGD	← DC_H →						
	同様の順序でD_2, D_1, DM を送出	P_4	← D_L →								
		P_5			← D_H →						
VECTE	グラフィックス描画開始	C	0	1	1	0	1	1	0	0	
TEXTW	グラフィックまたはテキスト・コード設定	C	0	1	1	1	1	← RA →			RA:データを書き始めるラスト・アドレス PTN:グラフィック描画時の線のビット・パターン TX_n:ドット構成データ
		P_1	← TX_8またはPTN_L →								
		P_2	← TX_7またはPTN_H →								
		⋮	⋮								
		P_8	← TX_1 →								
TEXTE	グラフィック/文字描画の実行開始	C	0	1	1	0	1	0	0	0	
CSRW	描画アドレス設定	C	0	1	0	0	1	0	0	1	EAD:描画開始ワード・アドレス dAD:描画開始ドット・アドレス
	文字描画時…P_1, P_2のみ 文字モード時…EAD_Mの上位3ビット0 文字/グラフィック・モード EAD_H=0	P_1	← EAD_L →								
		P_2	← EAD_M →								
		P_3	← dAD →				0	0	← EAD_H →		
CSRR	描画アドレス読み出し	C	1	1	1	0	0	0	0	0	CSRW コマンドに同じ
	読み出し専用	P_1	← EAD_L →								
		P_2	← EAD_M →								
		P_3							← EAD_H →		
		P_4	← dAD_L →								
		P_5	← dAD_H →								
MASK	マスク・レジスタ値の設定	C	0	1	0	0	1	0	1	0	MASK: マスキング/ドット・アドレス・レジスタ値
		P_1	← $MASK_L$ →								
		P_2	← $MASK_H$ →								

パラメータ	設定	機　　　　能
DE	0 1	表示停止 表示開始
CHR, G	0　0 1　0 0　1 1　1	文字/グラフィック混在モード 文字モード グラフィック・モード 禁止
F	0 1	フラッシュ描画 フラッシュレス描画
I, S	0　0 0　1 1　0 1　1	ノン・インターレース 禁止 インターレース インターレース・シュリンク
D	0 1	スタティック RAM 制御 ダイナミック RAM 制御(リフレッシュ動作)
C/R		1 行あたりの表示文字数設定(00H～FFH=2～256 文字/奇数)
HS		水平同期信号の幅の定義(00H～1FH=1～32 文字)
HFP		右方向の非表示区間の定義(00H～3FH=1～64 文字)
HBP		左方向の非表示区間の定義(00H～3FH=1～64 文字)
VSH+VSL		垂直同期信号の幅の定義(01H～1FH=1～31 ライン)
VFP		下方向の非表示区間の定義(01H～3FH=1～63 ライン)
VBP		上方向の非表示区間の定義(01H～3FH=1～63 ライン)
L/FH+L/FL		1 画面あたりの表示ライン数の設定 (000H～3FFH=1024, 1～1023 ライン)

◀〈図 2-81〉
SYNC

〈図 2-82〉 PC9801 シリーズの SYNC の設定値

パラメータ	専　用 高解像度	標準ディ スプレイ
CHR	0	←
F	1	←
I	0	←
D	0	←
G	0	←
S	0	←
HS	07H	←
HBP	07H	0DH
HFP	09H	←
C/R	4EH	←
VS	08H	←
VBP	19H	25H
VFP	07H	0FH
L/F	190H	C8H

▶ SYNC

表示動作モード等の定義．パラメータで非常に多くの設定ができます．ごく簡単ですが図 2-81 に紹介しておきます．GDC コマンド一覧と併せて見てください．図 2-82 は PC98 シリーズにおける設定です．

▶ MASTER/SLAVE

マスタ動作かスレーブ動作の選択をします．

マスタ=6FH/スレーブ=6EH

◈ GDC のコマンド(表示制御)

▶ START

表示開始の指示をします．

▶ STOP

表示停止の指示をします．

▶ ZOOM

表示時の拡大係数(ZR)，グラフィック文字の拡大係数(ZW)の設定です．

ZR/ZW：00H～0FH=1～16 倍

▶ SCROLL

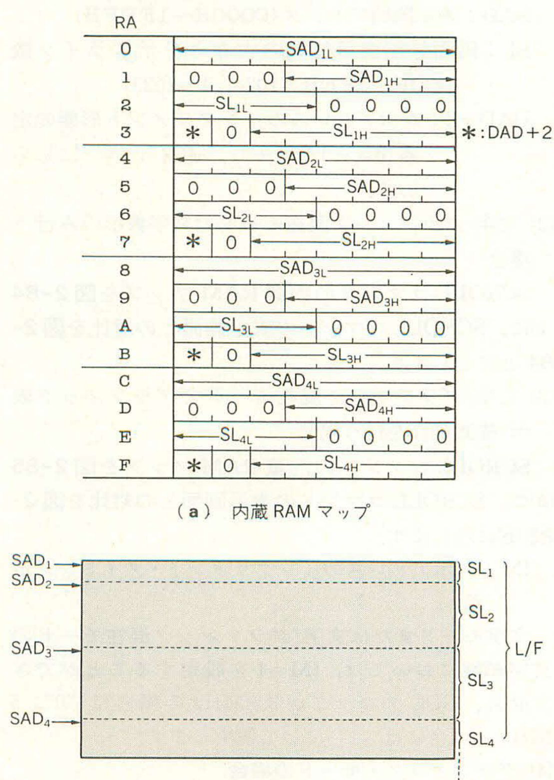
表示開始アドレス，画面分割表示領域の大きさの設定です．パラメータは GDC 内蔵の RAM に書いて，SCROLL コマンド実行時は RA(RAM アドレス)を設定するだけです．以下は，内蔵 RAM に送る各パラメータです．

RA：変更する内部 RAM の先頭アドレス(0～15)

(1) 文字モードの場合

図 2-83 (a)に SCROLL コマンドの内蔵 RAM マップを，図 2-83 (b)に SCROLL コマンドの表示画面との対比を示します．

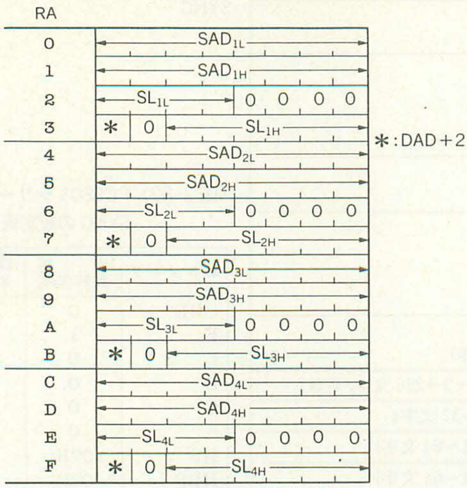
〈図 2-83〉 文字モードのスクロール・コマンド



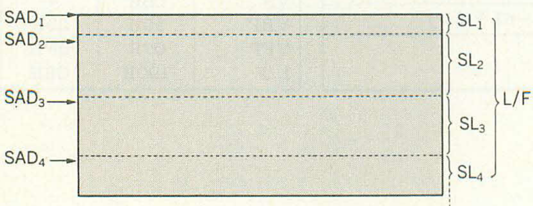
(a) 内蔵 RAM マップ

(b) 表示画面との対応

〈図 2-84〉 混在モードで文字表示のみ行う



(a) 内蔵 RAM マップ



(b) 表示画面との対応

SAD：表示開始アドレス(0000H～1FFFH)

SL：画面分割表示領域の大きさを示すライン数(000H～3FFH=1024，1～1023)

DAD+2：表示アドレスのインクリメント形態の定義(0=+1/1=+2)．通常は“0”にしておく．

(2) 文字/グラフィック混在モードで文字表示のみ行う場合

SCROLL コマンドの内蔵 RAM マップを図 2-84 (a)に，SCROLL コマンドの表示画面との対比を図 2-84 (b)に示します．

(3) 文字/グラフィック混在モードでグラフィック表示/描画動作を行う場合

SCROLL コマンドの内蔵 RAM マップを図 2-85 (a)に，SCROLL コマンドの表示画面との対比を図 2-85 (b)に示します．

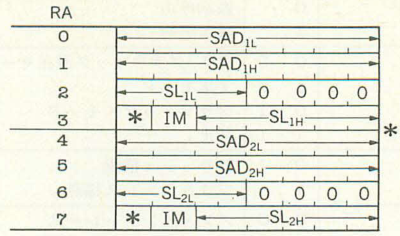
IM：表示アドレスのインクリメント・タイミング定義

文字モードまたは文字/グラフィック混在モードの文字領域においては，IM=1 を設定することができません．GDC クロックが 2.5 MHz の場合は“0”，5 MHz の場合には“1”にして使用します．

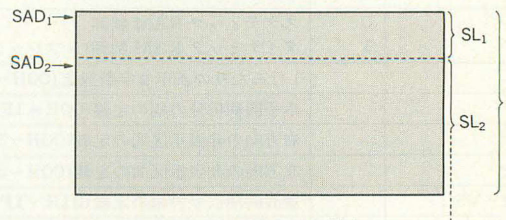
(4) グラフィック・モードの場合

SCROLL コマンドの内蔵 RAM マップを図 2-86

〈図 2-85〉 混在モードでグラフィック表示/描画動作を行う

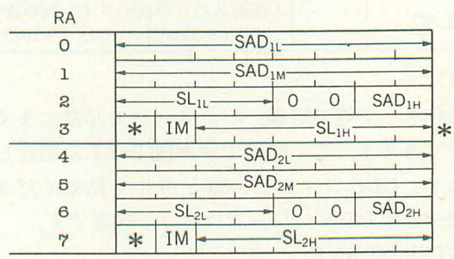


(a) 内蔵 RAM マップ

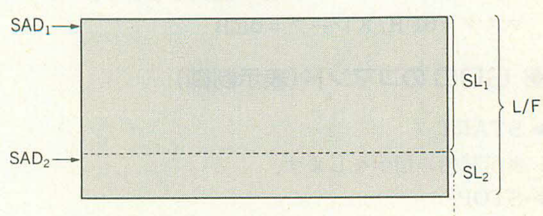


(b) 表示画面との対応

〈図 2-86〉 グラフィック・モード



(a) 内蔵 RAM マップ



(b) 表示画面との対応

(a)に，SCROLL コマンドの表示画面との対比を図 2-86 (b)に示します．

▶ CSRFORM

ライン・カウントの上限，文字表示時のカーソルの表示形態を定義します．

L/R：1 行中のライン数を定義する(00H～1FH=1～32 ライン)

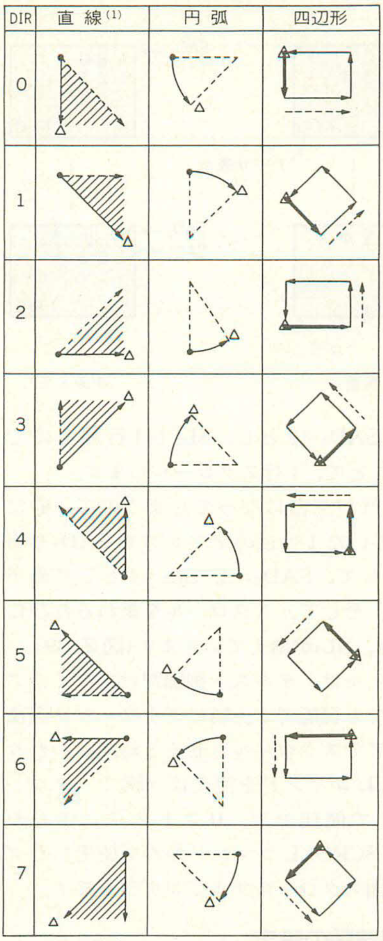
CS：カーソルの有無を定義する(0=カーソルなし/1=カーソルあり)

BD：カーソルの点滅の有無を定義する(0=点滅する/1=常時点灯)

BL：カーソルの点滅周期の定義(01H～3FH=4～124)

CST：カーソルの表示開始ラインの定義(00H～1FH=1～32 ライン)

〈図 2-87〉VECTW



● :描画始点　━━▶ :第 1 方向　▨ :定義域
─→ :描画方向　- - -→ :第 2 方向
△ :描画終了時の EAD 指示点

CF₁：カーソルの表示終了ラインの定義
　　(OOH～1FH＝1～32 ライン)

▶ PITCH

映像メモリの横幅を定義します．これは表示領域(C/R)とは別で，表示されない部分を含めた映像メモリの横幅です．スクロール等で使用します．SYNC コマンドを設定すると C/R の値で初期化されてしまうので，SYNC コマンドの後に使用します．

P_1：OOH～FFH＝0～255(文字数/行)

▶ LPEN

ライト・ペンで得られたアドレスを出力します．出力は 3 バイトです．

D_1：LAD_L(8 ビット)

D_2：LAD_M(8 ビット)

D_3：LAD_H(2 ビット)

◈ GDC のコマンド(描画制御)

主にグラフィック画面の描画用のコマンドです．比較的高機能な描画コマンドを持っていますが，最近では CPU が高速化され，GDC で描画するより CPU で描画したほうが速い場合もあります．

〈図 2-88〉描画時のアドレスの進み方

	DC	D	D_2	D_1	DM
初期値	0	8	8	−1	−1
直　線	$\|\triangle X\|$	$2\|\triangle Y\|-\|\triangle X\|$	$2\|\triangle Y\|-2\|\triangle X\|$	$2\|\triangle Y\|$	—
円	$(r/\sqrt{2})\uparrow$	$r-1$	$2(r-1)$	−1	0
弧	N	$r-1$	$2(r-1)$	−1	M
四辺形	3	A*	B*	−1	A*

△X：X 座標変位，r：半径，M：マスキング・ドット数，
△Y：Y 座標変位，N：描画総ドット数，↑：切り上げ，
A*：第 1 描画方向変位数，B*：第 2 描画方向変位数

▶ VECTW

描画方向(8 方向)，描画種類(直線/円弧/四辺形/文字)，描画用パラメータ(XY 軸/半径)等を定義します．円は一度に 1/8 しか描画されません．

SL，R，C，T，L：描画種類

　　直線　＝0，0，0，0，1

　　円弧　＝0，0，1，0，0

　　四辺形＝0，1，0，0，0

DIR：描画方向[下向きが「0」で，左回りに 45 度単位で 8 方向(図 2-87)]

DGD：描画時のアドレスの進み方(GDC クロック：2.5 MHz＝0/5 MHz＝1)

DC，D，D_2，D_1，DM：描画パラメータ(図 2-88)

参考に描画制御プログラム GDC.C をリスト 2-6 に示します．

▶ VECTE

直線，四辺形，円弧，1 ドット描画の実行開始を指示します．

▶ TEXTW

実線，破線等の線種データ，グラフィック文字用のドット構成データを設定します．

これらのデータは内部 RAM に送ります．

RA：内部 RAM の先頭アドレス(0～7)

線種データは 2 バイト，グラフィック文字用のドット・データは 8 バイトです．

▶ TEXTE

グラフィック文字描画の実行開始を指示します．

▶ CSRW

描画実行ワード・アドレス(EAD)や，描画実行ドット・アドレス(dAD)を設定します．文字制御時にはカーソル位置を，グラフィック制御時には描画開始点を定義します．

▶ SCRR

描画実行ワード・アドレス(EAD)や，描画実行ドット・アドレス(dAD)を読み出します．読み出されるデータは 5 バイトです．

▶ MASK

MASK レジスタの値を設定します．

◈ GDC のコマンド（メモリ制御）

▶ WRITE

ドット修正モードや，映像メモリにデータ（このコマンドの後に続くパラメータ）を書き込むための準備をします．WRITE コマンドを実行する前に，書き込みたいブロック（WLH による）数−1と，書き込み開始アドレスを，VECTW，CSRW コマンドによって設定します．ドット修正モードだけ変更したい場合は必要ありません．

WLH： 00＝ワード転送
01＝禁止
10＝下位バイト転送（上位バイトは0）
11＝上位バイト転送（下位バイトは0）

MOD： ドット修正モードの選択
00＝REPLACE（上書）
01＝COMPLEMENT（反転）
10＝CLEAR（0 を描く）
11＝SET（1 を描く）

▶ READ

映像メモリの内容を読み出すコマンドです．あらかじめ読み込みたいブロック（WLH による）数−1と読み込み開始アドレスを，VECTW，CSRW コマンドによって設定します．

WLH：WRITE コマンドと同じ
MOD：WRITE コマンドと同じ

▶ DMAW

DMA を使用した WRITE コマンドです．使用方法は WRITE コマンドと同じです．PC98 では DMA は使用できません．

▶ DMAR

DMA を使用した READ コマンドです．使用方法は READ コマンドと同じです．PC98 では DMA は使用できません．

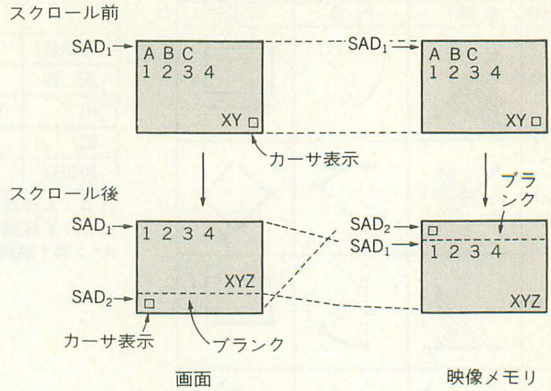
◈ スクロール，画面分割の方法

SCROLL コマンドによって設定する SAD（表示開始アドレス），SL（表示領域のライン数）により，表示画面を最大4個（グラフィック画面は2個）まで分割して表示できます．

SAD，SL は，SCROLL コマンドによって，GDC 内蔵 RAM にいったん格納されます．GDC が画面出力の際に，この RAM を見ながら出力します．

SAD，SL を少しずつ変化させることで，表示位置を少しずつ変えてスクロールさせることができます．例えば SAD_1＝1行，SL_1＝25行で画面全体が表示されているときに，1行スクロールすると最上行は消え，最下位行に新しい行が見えてきます．ここで SAD_1 を

〈図 2-89〉ページ内スクロール



1行繰り上げて SAD_1＝2とし，SL_1 も1行減るので SL_1＝24 とすることで，1行スクロールします．

これでは最下行が空白になってしまうので，前に SAD_1 で使用していた1行目のアドレスを SAD_2 で新たな表示画面として，SAD_2＝2，SL_2＝1として最下行に表示します．そして，スクロールを重ねるたびに SL_1 は減っていき，SL_2 が増えていきます（図 2-89）．

画面のスクロールは，テキスト画面だけでなく，グラフィック画面でも可能です．特にグラフィック画面は，ソフトウェアでスクロールさせると時間がかかりますが，SCROLL コマンドを使えば一瞬でスクロールが完了するので便利です．リスト 2-7 に示した SCROLL.C は，SCROLL コマンドのみを使用したグラフィックを画面スクロールするプログラムです．

◈ クロック周波数の設定

GDC に与えるクロック周波数は1水平走査期間内の表示時間 C/R と，そのときの映像メモリ・アクセス数（水平方向表示ドット数/1ワード当たりのビット数）によって決定されます．

計算方法は以下のとおりになります．

HFP＋HS＋HBP＋C/R＝水平走査期間
C/R/16＝1回のメモリ・アクセスに要する時間
HS＝水平同期信号の幅
HFP＝右方向の非表示区間
HBP＝左方向の非表示区間
C/R＝1行あたりの表示時間

表示サイクルは DAD＋2 モードを使用しない通常表示では2クロックですので，1回のメモリ・アクセスに要する時間の半分のクロックを GDC に入力します．

PC98 のノーマル・モードで，専用高解像度 CRT（400 ライン）の水平走査周波数は，24.83 kHz（40.28 μs）です．GDC に与えるパラメータは，HFP＝07H（3.04 μs），HS＝07H（3.04 μs），HBP＝09H（3.8 μs）となり，水平方向ドット数は 640 ドットで，その表示

〈リスト 2-7〉
スクロール・コマンド

```
/*
**  GDC (SCROLL) のテスト
**
**  グラフィック画面の垂直方向1ドットスクロール
**
*/

#include <dos.h>

#define     GDC_STAT    0xa0
#define     GDC_CMD     0xa2
#define     GDC_PARA    0xa0

void main(void)
{
    int         i,j,im;

    im = (peekb(0,0x054d)&0x04)<<4;             /* GDCｸﾛｯｸで IMを設定  */

    for (i = 1; i < 400; i++){
        while ((inportb(GDC_STAT)&0x04) == 0);  /* FIFO が空くまで待つ */
        outportb(GDC_CMD,0x70);                 /* スクロールコマンド  */
        outportb(GDC_PARA,(i*40)&0xff);         /* SAD1 LOW            */
        outportb(GDC_PARA,(i*40)>>8);           /* SAD1 HI             */
        outportb(GDC_PARA,((400-i)<<4)&0xff);   /* SL1 LOW             */
        outportb(GDC_PARA,im|((400-i)>>4));     /* DAD=0, SL1 HI       */
        outportb(GDC_PARA,0);                   /* SAD2 LOW            */
        outportb(GDC_PARA,0);                   /* SAD2 HI             */
        outportb(GDC_PARA,(i<<4)&0xff);         /* SL2 LOW             */
        outportb(GDC_PARA,im|(i>>4));           /* DAD=0, SL1 HI       */
        for (j = 0; j < 30000; j++);
    }
}
```

〈図 2-90〉
ディスプレイ・タイミング

●ノーマル・モード

Symbol	専用高解像度ディスプレイ	標準ディスプレイ
H	40.28 μs (24.83 kHz)	62.58 μs (15.98 kHz)
HDISP	30.4 μs	44.70 μs
HFP	3.04 μs	4.47 μs
HS	3.04 μs	4.47 μs
HBP	3.8 μs	8.94 μs
V	17.72 ms (440 H, 56.4 Hz)	16.33 ms (261 H, 61.2 Hz)
VDISP	16.11 ms (400 H)	12.52 ms (200 H)
VFP	0.28 ms (7 H)	0.94 ms (15 H)
VS	0.32 ms (8 H)	0.50 ms (8 H)
VBP	1.01 ms (25 H)	2.38 ms (38 H)

●ハイレゾリューション・モード

Symbol	タイミング	
	奇数フィールド	偶数フィールド
H	30.45 μs (32.84 kHz)	
HDISP	23.41 μs	
HFP	2.34 μs	
HS	1.76 μs	
HBP	2.93 μs	
V	12.5 ms (410.5 H, 80 Hz)	
VDISP	11.42 ms (375 H)	
VFP	0.244 ms (8H)	0.259 ms (8.5 H)
VS	0.152 ms (5 H)	
VBP	0.685 ms (22.5 H)	0.670 ms (22 H)

時間はC/R＝4EH・HDISP＝30.4 μsです．これからメモリ・アクセス時間は0.76 μsで，GDCのクロックは2.63 MHz(通称2.5 MHz)となります．標準CRT(200ライン)では1.79 MHzのクロックがかかります．

GDCのSYNCコマンドでは，これらのパラメータを定義してタイミングを作ることができます(SYNCコマンドのパラメータは時間指定ではなく文字数で指定する)．専用高解像度モードであっても，640×400ドット以外の画面を出せますが，GDCに与えられているクロックは固定されていますので，自由度は少なくなっています．

図2-90にディスプレイ・タイミングを示します．

◈ GDCのクロック

PC9801シリーズでは，GDCは2.5 MHzで動作するのが基本になっていますが，5 MHzに切り替えることも可能です．GDCのクロックを倍(IMAGEモード)にすることで描画速度は速くなりますが，GDCに対して送るコマンドが若干異なってきます．反対に，計算上必要なクロックの半分をGDCに与えるモード(DAD+2モード)もありますが使用しません．これらはSCROLLコマンドで設定します．

GDCの動作クロックは，システム共通領域(0000：054DH・ビット3)が“1”のときに5 MHz，“0”のときに2.5 MHzになります．5 MHz時には，SCROLLコマンドではIM＝1，VECTWコマンドではDGD＝1にして使用します．

◈ GDCのバージョンの違い

PC98シリーズでは，

μPD7220	PC9801FA以前の機種
μPD7220A	PC9801US
μPD72020	PC9801FA以降の機種

のように，その発売時期によって新しいGDCが使用されています．これらは上位互換で，新しくなるたびに少しずつ機能が増えています．

〈リスト2-6〉描画制御コマンド

```
/*
**  ＧＤＣテストプログラム
*/

#include <dos.h>

#define     GDC_STAT    0xa0
#define     GDC_CMD     0xa2
#define     GDC_PARA    0xa0

#define     VECTW       0x4c
#define     VECTE       0x6c
#define     CSRW        0x49
#define     TEXTW       0x78
#define     START       0x6b

#define     VRAM_ON     0
#define     VRAM_OFF    1
#define     BIOS_CRT    0x18

    typedef struct {
        int             slrctldir;  /*  描画種類／描画方向      */
        unsigned int    dc;
        unsigned int    d;
        unsigned int    d2;
        unsigned int    d1;
        unsigned int    dm;
    } vectwpara;

    int     gdcclock;

void setlinestyle(unsigned int pattern);                /*  線種の設定  */
void line(int x1,int y1,int x2,int y2,int plane);       /*  直線        */
void box(int x1,int y1, int x2,int y2,int plane);       /*  長方形      */
void circle(int x,int y,int radius,int plane);          /*  円          */

void godraw(int x,int y,int plane,vectwpara *vp);
void outgdccmd(int data);
void outgdcpara(unsigned int data,int len);
void outgdc(int port,int data);
void wait(void);

void crtinit(void);

int main()
{
    int     i;

    crtinit();
    setlinestyle(0xffff);                               /*  線種－＞実線    */

    for (i = 10; i < 640; i+=20){
        line(320,200,i,0,0);
        line(320,200,i,399,0);
    }
    for (i = 10; i < 400; i+=20){
        line(320,200,0,i,0);
        line(320,200,639,i,0);
    }

    for (i = 0; i < 150; i+=10){
        box(i,i,639-i,399-i,1);
    }

    for (i = 1; i < 150; i+=5){
        circle(320,200,i,2);
    }
    return 0;
}

/*
**  描画する線種を設定する。描画モードはリプレイス
*/
void setlinestyle(unsigned int pattern)
{

    outgdccmd(0x20);    /*  WRITE  WLH=00(ワード転送) MOD=00(REPLACE)     */

    outgdccmd(TEXTW);   /*  TEXTW  線種データの設定 */
    outgdcpara(pattern,2);
}

/*
**  (x1,y1) から (x2,y2) を結ぶ直線を plane のＶＲＡＭプレーンに描画する
*/
void line(int x1,int y1,int x2,int y2,int plane)
{
    int         dir,dmy,dx,dy;
    vectwpara   vp;

/*  描画方向(dir)が０～３になるように座標を交換する */
    if ((x1 > x2)||((x1 == x2)&&(y1 > y2))){
        dmy = x1; x1 = x2; x2 = dmy;
        dmy = y1; y1 = y2; y2 = dmy;
    }
    dx = x2-x1; dy = y2-y1;

/*  座標から dir,dx,dy を求める */
    if (dy > 0){
        if (dx < dy){
            dir = 0;
            dmy = dx; dx = dy; dy = dmy;    /*  dir=0 は dx dy を交換する   */
        }else{
            dir = 1;
        }
    }else{
        dy = -dy;
        if (dx > dy){
            dir = 2;
        }else{
            dir = 3;
            dmy = dx; dx = dy; dy = dmy;    /*  dir=3 は dx dy を交換する   */
        }
    }
    vp.slrctldir = 0x08|dir;                /*  直線    */
    vp.dc = dx; vp.d1 = 2*dy; vp.d = vp.d1-dx; vp.d2 = vp.d-dx; vp.dm = 0;

    godraw(x1,y1,plane,&vp);
}

/*
**  (x1,y1)を左上、(x2,y2)を左下とする長方形を plane のＶＲＡＭに描画する。
*/
void box(int x1,int y1, int x2,int y2,int plane)
{
    int         dx,dy,dmy,dir;
    vectwpara   vp;

/*  左上が(x1,y1) 右下が(x2,y2)になるように座標を交換する。 */
    if (x1 > x2){
```

```
        dmy = x1; x1 = x2; x2 = dmy;
    }
    if (y1 > y2){
        dmy = y1; y1 = y2; y2 = dmy;
    }
    dx = x2-x1; dy = y2-y1;

    dir = 0;                        /*  ＤＩＲは０固定       */
    vp.slrctldir = 0x40|dir;        /*  四辺形              */
    vp.dc = 3; vp.d1 = -1; vp.d = dy; vp.d2 = dx; vp.dm = dy;

    godraw(x1,y1,plane,&vp);
}

/*
**  中心座標(x,y) 半径 radius の円を plane のＶＲＡＭに描画する。
*/
void circle(int x,int y,int radius,int plane)
{
    int         dir;
    vectwpara   vp;
    int         xx[8],yy[8];

/*  ＤＩＲの変化による描画開始点を設定  */
    xx[0] = xx[3] = x-radius; yy[0] = yy[3] = y;
    xx[1] = xx[6] = x; yy[1] = yy[6] = y-radius;
    xx[2] = xx[5] = x; yy[2] = yy[5] = y+radius;
    xx[4] = xx[7] = x+radius; yy[4] = yy[7] = y;

    vp.dc = (int)((long)radius*10000/14142)+1;      /*  radius/√２↑     */
    vp.d = radius-1; vp.d2 = 2*vp.d; vp.d1 = -1; vp.dm = 0;

    for (dir = 0; dir < 8; dir++){
        vp.slrctldir = 0x20|dir;                    /*  円      */
        godraw(xx[dir],yy[dir],plane,&vp);
    }
}

/*
**  ＶＥＣＴＷ，ＣＳＲＷ，ＶＥＣＴＥ
*/
void godraw(int x,int y,int plane,vectwpara *vp)
{
    unsigned int    ead;

    outgdccmd(CSRW);                                /*  描画開始位置の設定  */
    ead = (unsigned int)((plane+1)%4)*0x4000+y*40+x/16;
    outgdcpara(ead,2);
    outgdcpara((x%16)<<4,1);

    outgdccmd(VECTW);                               /*  ＶＥＣＴＷ  */
    outgdcpara(vp->slrctldir,1);
    outgdcpara((vp->dc)|(gdcclock<<14),2);          /*  DGD=1:5MHz  =0:2.5MHz   */
    outgdcpara(vp->d,2);
    outgdcpara(vp->d2,2);
    outgdcpara(vp->d1,2);
    outgdcpara(vp->dm,2);

    outgdccmd(VECTE);                               /*  描画開始    */

    while ((inportb(GDC_STAT)&0x08) != 0)   wait(); /*  描画終了まで待つ    */
}

/*
**  ＧＤＣにコマンドを設定する。
*/
void outgdccmd(int dat)
{
    outgdc(GDC_CMD,dat);
}

/*
**  ＧＤＣにパラメータを送る。 len 1=BYTE 2=WORD
*/
void outgdcpara(unsigned int dat,int len)
{
    int     i;

    for (i = 0; i < len; i++){
        outgdc(GDC_PARA,dat&0xff);
        dat >>= 8;
    }
}

/*
**  Ｇ－ＧＤＣに対して１バイトのデータを出力
*/
void outgdc(int portadr,int dat)
{
    wait();                                             /*  GDC wait    */
    while ((inportb(GDC_STAT)&0x02) != 0)   wait();     /*  FIFO check  */
    outportb(portadr,dat);
}

/*
**  ＷＡＩＴ
*/
void wait(void)
{
    outportb(0x5f,0);
    outportb(0x5f,0);
}

/*
**  ８色  ４００ライン
**  描画ＶＲＡＭ表  表示ＶＲＡＭ表
*/
void crtinit(void)
{
    union REGS  regs;

    regs.h.ah = 0x42;
    regs.h.ch = 0xc0;

    int86(BIOS_CRT,&regs,&regs);    /*  ４００ライン，カラー，  */
    int86(BIOS_CRT,&regs,&regs);

    outportb(0x6a,0);               /*  ８色モード              */
    wait();
    outportb(0xa4,0);               /*  表示画面  ＶＲＡＭ表    */
    wait();
    outportb(0xa6,0);               /*  描画画面  ＶＲＡＭ表    */
    wait();
    outportb(GDC_CMD,START);        /*  表示開始                */

    gdcclock = (peekb(0,0x054d)&0x04)>>2;       /*  gdcclock 1=5M  0=2.5M   */
}
```

フロッピ・ディスク・インターフェース

▶使用 LSI
μPD765A 相当
<1MB 用フロッピ・ディスク・インターフェース>
▶ I/O アドレス
90H, 92H(FDC)
94H(ライト・コントロール/リード・シグナル)
▶使用割り込み
IR_{11}
▶使用 DMA
2
<640KB用フロッピ・ディスク・インターフェース>
▶ I/O アドレス
C8H, CAH(FDC)
CCH(ライト・コントロール/リード・シグナル)
▶使用割り込み
IR_{10}
▶使用 DMA
3
<640KB/1MB インターフェース切り替え>
▶ I/O アドレス
BEH

PC9801で使用されるフロッピ・ディスク・ドライブには，8インチ2D(1MB)，5インチ2D(320KB)，5インチ2DD(640KB)，5インチ2HD(1MB)，5インチ640KB/1MB両用型，3.5インチ2DD(640KB)，3.5インチ2HD(1MB)，3.5インチ640KB/1MB両用型が使われています．

フロッピ・ディスク・インターフェースには，320K，640KB，1MB，本体内蔵の640KB/1MB両用タイプがあります．また，最近の機種ではPC/AT互換機で標準の3.5インチ2HD(1.44MB)のメディアを扱うこともできます．

◈ 5インチ2D(320KB)フロッピ・ディスク・インターフェース

5インチ2D型はPC8001，PC8801等で使用されていた外部接続用のインテリジェント型のフロッピ・ディスク・ドライブ・ユニット(PC80S31等)をパラレル・インターフェース(8255)経由でアクセスできましたが，PC9801M以降の機種には，このインターフェースは付いていません．

しかし，640KB/1MB両用型のドライブを持つ機種では，BIOSレベルで2Dを仮想的に読み書きすることが可能です．ただし，ドライブの構造的問題で書き込みは保証されません．

◈ 640KB・1MB専用フロッピ・ディスク・インターフェース

640KBおよび1MBのフロッピ・ディスク・インターフェースには，FDC(フロッピ・ディスク・コントローラ)としてμPD765Aの相当品(以降765と略す)を使用しています．FDCのデータ転送にはDMAを使用します．

640KBと1MBでは，使用する割り込みレベル，DMAチャネル，I/Oポートが別になっていて，基本的には独立したFDCを持つ構造になっており，それぞれ4台ずつのFDD(フロッピ・ディスク・ドライブ)を付けることが可能です．

PC9801VM/UV以降の機種にはすべて，640KB/1MB両用型のフロッピ・ディスク・インターフェースが付いていますから，拡張スロットに640KB，1MB専用のフロッピ・ディスク・インターフェースを付ける必要はなくなっています．別に専用インターフェースを付けたい場合は，本体の両用型インターフェースをどちらかのモードに固定して使うことになります．

◈ 640KB/1MB両用型フロッピ・ディスク・インターフェース

640KB/1MB両用型のインターフェースは，640KB，1MB用のFDCを共用しています．しかし，割り込み，DMA，I/Oアドレス等は，それぞれの専用インターフェース同様に独立して割り当てられていて，モード切り替えポート(0BEH)で切り替えることができます．

通常，BIOSで使用する場合は，640KB専用モードと，640KB/1MB自動切り替えモードがあり，ディップ・スイッチ(SW_{3-1}/SW_{3-2})で切り替えます．640KB専用モードでは，640KB専用インターフェースと同様のDMAチャネル(#3)，割り込みレベル(IR_{10})を使用し，640KB/1MB自動切り替えモードでは，1MB専用インターフェースと同様のDMAチャネル(#2)，割り込みレベル(IR_{11})を使用します．

640KB/1MB両用のタイプのFDD，フロッピ・ディスク・インターフェースが搭載されている機種でも，仕様では，外部に増設するFDDは1MB専用になっています．これは，増設用FDDコネクタに640KB/1MB切り替え出力が出ていないせいです．増設用FDDに640KB/1MB両用型を使い，増設用FDDユニットの2ピンと，内蔵FDDユニットの2ピンを接続すれば，増設用FDDも640KB/1MBの切り替えができます(ただし，MS-DOSの一部のコマンドは自動切り替えに対応できない)．

また，FDDのストラップを変更しFDD単体で自動切り替えできるモードにし，フロッピ・ディスク・

〈図 2-91〉モード・チェンジ・レジスタ

命令	I/O ポート・アドレス	R/W	データ D7	D6	D5	D4	D3	D2	D1	D0	備考
ライト・モード・チェンジ	00BE	W	0	0	0	0	0	EMTON	FDD EXC	PORT EXC	μPD765A 外部レジスタ．初期値は 00
リード・モード・ステータス	00BE	R	×	×	×	×	DSW	FIX	FDD EXC	PORT EXC	

BIOS に細工をすることで 640KB/1MB 自動切り替えにすることも可能です．フリー・ソフトウェアでは「DUALDRV.SYS」(T. Haraikawa 氏製作)等があります．

◈ モード・チェンジ・レジスタ

640KB/1MB 両用型フロッピ・ディスク・インターフェースの切り替えを行うポートです．PORT EXC ではインターフェースの I/O アドレス，DMA チャネル，割り込みレベル等の CPU に密接する部分の切り替えを行います．FDD EXC は，FDC の動作モードを変えるもので，VFO パラメータや FDC クロック等の切り替えを行います．

640KB/1MB 自動切り替えモードでは，PORT EXC をどちらかに固定し，FDD EXC を実際に読める(エラーが起きない)ほうに切り替えることで，両方のメディアを扱っています．通常では PORT EXC は電源投入時の初期設定で固定化され，動作途中に変更されることはありません(図 2-91)．

▶ D_0：PORT EXC

インターフェースの I/O アドレスを設定します．

1：1MB インターフェース用(90H，92H，94H)

0：640KB インターフェース用(C8H，CBH，CCH)

▶ D_1：FDD EXC

VFO パラメータと FDC のクロックを設定します．

1：1MB モード

0：640KB モード

▶ D_2：EMTON

FDD のモータ制御用．EMTON=1 で 1MB モード時に MTON の設定を有効にします．EMTON=0 で 1MB モード時に強制的に，常にモータ ON の状態にします．

◈ モード・チェンジ・レジスタ(BEH/リード)

▶ D_0：PORT EXC

インターフェースの I/O アドレス設定状態を読み出します．

1：1MB インターフェース用(90H，92H，94H)

0：640KB インターフェース用(C8H，CBH，CCH)

▶ D_1：FDD EXC

VFO パラメータと FDC のクロック設定状態を読み出します．

1：1MB モード

0：640KB モード

▶ D_2：FIX

ディップ・スイッチ SW_{3-1}の状態を示します．

1：PORT EXC を無効とする固定モード

0：有効とする自動切替えモード

▶ D_3：DSW

ディップ・スイッチ SW_{3-2}の状態を示します．

FIX=0 の場合に FDD の立ち上がり状態を読み取り，FIX=1 の場合に固定するモードを示します．

1：1MB モード

0：640KB モード

▶ D_6：TMF(EPSON-PC シリーズ)

1MB インターフェース使用時

0：FDD モータ停止

1：FDD モータ回転中

640KB インターフェース使用時

0：ワンショット・タイマ動作中

1：ワンショット・タイマ，タイム・アウト

▶ D_7：IFDC(EPSON-PC シリーズ)

0：FDC の割り込みがある

1：FDC の割り込みがない

◈ μPD765A(FDC)

FDC には二つの I/O アドレスがあり，ステータス・レジスタもリードとデータ・レジスタの読み書きがあります．データ・レジスタはコマンド，パラメータ，データ，リザルト・ステータスがあります．

● ステータス・レジスタ(90H，C8H/リード)

▶ステータス読み出し

ステータス・レジスタではドライブの状態，FDC の状態等が読み出せます．

● データ・レジスタ(92H，CBH/リード&ライト)

▶コマンド設定(Command Phase)

「コマンド」の設定は，FDC がアイドリング状態のときにデータ・レジスタに書き込むことで実行します．パラメータが必要な場合はコマンドに続いてデータ・レジスタに書き込みます．

▶コマンド実行(Execution Phase)

必要なパラメータの送出が終わったら，FDC は Execution Phase に入り，与えられたコマンドを実行します．必要なら DMA 等を通してメモリに対して

〈図 2-92〉ライト・コントロールの I/O アドレス

	命令	I/O ポート・アドレス	R/W	データ								備考
				D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
FDC μPD765	ライト・コマンド	92 CA	W	← コマンド・レジスタ →								μPD765A へコマンドをセットする
	ライト・レジスタ	92 CA	W	← パラメータ →								μPD765A へのパラメータをセットする．NON-DMA モード時は FD への書き込みデータもセットする
	リード・ステータス	90 C8	R	← ステータス・レジスタ →								μPD765A からステータスを引き取る
	リード・データ	92 CA	R	← リザルト・ステータス →								μPD765A からリザルト・ステータスを引き取る．NON-DMA モード時には FD から読み取ったデータも引き取る
コントローラ	ライト・コントロール	94	W	RST	FRY	×	×	×	×	×	×	
		CC	W	RST	FRY	AIE	DMAE	MTON	TMSK	×	TTRG	
	リード・スイッチ/シグナル	94	R	$FINT_1$	$FINT_0$	DMACH	0	TYP_1	TYP_0	0	0	$TYPE_1$ FDD＃3/＃4 $TYPE_0$ FDD＃1/＃2
		CC	R	$FINT_1$	$FINT_0$	DMACH	RDY	TYP_1	TYP_0	0	0	0：1 MB FDD 1：両用 FDD

上段 1MB インターフェース．下段 640KB インターフェース
PC9801VF2/VM0,2,4/UV2/VM21/VX0,2,4/UV21/VX01,21,41 では，EMTON ビットは無効

読み書きします．

▶結果出力(Result Phase)

最後は，コマンドの実行結果を報告するためのリザルト・ステータス情報をセットし，CPU がそれらの情報をデータ・レジスタを通して読み出します．必ず INT 要求によって処理されますので，すべてのリザルト・ステータス，パラメータ情報を指定された順序に従って読み取る必要があります．

◈ ライト・コントロール(94H, CCH/ライト)

94H＝1MB インターフェース用，
CCH＝640KB インターフェース用

図 2-90 に I/O アドレスを示します．

▶ D_0：TTRG(640KB のみ)

VFO の TRIG　IN 端子の入力信号であり，ディスク・ドライブのモータ ON/OFF 制御の時間設定用タイマのトリガとして使用します．TMSK＝1，TTRG＝1 とすると，約 100 ms 後，割り込み信号が発生します．このタイマ機能は，トリガ入力後，100 ms 以内に再びトリガを入力すれば，後のトリガが有効になります．

EPSON-PC シリーズでは 1MB でも有効ですので，TTRG＝0 にします．

▶ D_2：TMSK(640KB のみ)

FDC からのタイマ割り込みをマスクするレジスタで，TMSK＝0 で FDC からのタイマ割り込みが禁止されます．電源投入直後には「0」にしておきます．「1」にしてからタイマをトリガするとタイム・アウト以前に割り込み信号が 1 回出るので，注意する必要があります．

EPSON-PC シリーズでは 1MB でも有効ですので，TMSK＝0 にします．

▶ D_3：MTON(640KB のみ)

MTON＝0 で FDD のモータを同時にすべて停止させます．

▶ D_4：DMAE(PC9801 では無効)

DMA チャネルを使ってデータ転送を行うとき，DMA コントローラからの DRQ 信号，DACK 信号の許可/不許可を指示します．DMA 使用時は DMAE＝1 とします．

▶ D_5：AIE(640KB/1MB 両用型と PC9801U のみ)

AIE＝0 で FRY の入力を無視し，前の状態のままにします．

▶ D_6：FRY(640KB 専用にはなく，PC9801U にはある)

〈図 2-93〉リード・シグナル

VF

TYP_1	TYP_0	種別
0	0	外付け1MBのみ
1	0	内蔵両用
0	1	外付け1MB

VM他

TYP_1	TYP_0	種別
1	0	内蔵両用
0	1	外付け1MB

765のREADY端子に，ディスク・ドライブのRDY信号と論理和(AND)してつながっていて，ドライブの接続状態やドライブの電源投入状態をチェックするのに使用されます．

これらのチェックをするために，ドライブにキャリブレート動作をさせて，Track00信号が返ってくるか調べますが，765のREADY端子がOFFであると，キャリブレート動作がされず判別できないため，FRY=1で強制的にREADY端子をONにして調べます．通常動作のときは，FRY=0としておかないと，Not Readyを検出できなくなります．

▶ D_7：RST

765のRESET端子につながっていて，765を初期化するのに使用します．初期化は765へのコマンド，パラメータの転送シーケンスや，リザルト・ステータス転送シーケンスが乱れたときに使用します．なお，初期化は電源投入直後やRESETスイッチを押したときに，ハードウェアで行われます．RST=1で初期化されます．

◈ リード・シグナル(94H，CCH/リード)

94H=1MBインターフェース用，
CCH=640KBインターフェース用

▶ D_2〜D_3：TYP_0〜TYP_1(640KB/1MB両用型のみ)

使用できる4台のFDDの種別(640KB/1MB両用型，1MB専用型)を読み出せます(図2-93)．

▶ D_4：RDY(640KBのみ)

FDDのRDY端子の状態を読み取ります．RDY=1でReadyです．ドライブの有無のチェックはFRYの説明のように行います．このビットは，通常のシークや読み書きコマンドを実行する場合に，FDDのReadyをチェックするためのものです．

PC9801Uを除く640KB専用機種，インターフェース・ボードは，765のREADY端子は「1」で固定されています．

PC9801の94H・D_4は，プリンタ・インターフェースのPSTBに使用しています．他機種ではPSTBは37Hにあります．

▶ D_5：DMACH

640KBまた1MBの専用インターフェースでは，基板上のディップ・スイッチの内容が読み出せます．システム既定値や640KB/1MB両用機種では，1MB=0，640KB=1が読み出せます．

▶ D_6〜D_7：$FINT_0$〜$FINT_1$

640KBまたは1MBの専用インターフェースでは，基板上のディップ・スイッチの内容が読み出せます．システム既定値や，640KB/1MB両用機種では，$FINT_0$=1，$FINT_1$=0が読み出せます．

ハード・ディスク・インターフェース

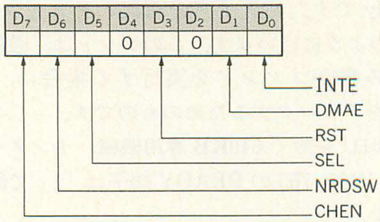
PC98シリーズで使用されるハード・ディスク・ドライブ(HDD)には，SASI，SCSI，IDEの3種類があります．SASIは古くからPC98シリーズに使用されてきた方式で，2ドライブ・合計80Mバイトまでしかサポートされていません．SCSIは近年，SASIの代わりに盛んに使われるようになった方式で，7台までのユニットを制御でき，CD-ROMドライブやMOドライブ，記憶装置以外のユニットも制御できます．

IDEは，PC/AT互換機等で使用されているHDDで，インターフェースがHDD内部に付いているために，基本的にはバスに直結する形をとります．また，

〈図 2-94〉SASIインターフェースのアドレス

命令	I/Oポート・アドレス	R/W	データ D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	備考
ODR　アウトプット・データ・レジスタ	80	W	OD_7	OD_6	OD_5	OD_4	OD_3	OD_2	OD_1	OD_0	
IDR　インプット・データ・レジスタ	80	R	ID_7	ID_6	ID_5	ID_4	ID_3	ID_2	ID_1	ID_0	
OCR　アウトプット・コントロール・レジスタ	82	W	—	NRDSW	SEL	0	RST	0	DMAE	INTE	内蔵タイプ
			CHEN	NRDSW	SEL	0	RST	0	DMAE	INTE	外付けタイプ
ISR　インプット・ステータス・レジスタ	82	R	REQ	0	BSY	MSG	CXD	IXO	—	INT	内蔵タイプ NRDSW≒1
			REQ	ACK	BSY	MSG	CXD	IXO	—	INT	外付けタイプ NRDSW=1
			CT_1	CT_0	DT_{02}	DT_{01}	DT_{00}	DT_{12}	DT_{11}	DT_{10}	NRDSW=0

〈図 2-95〉OCR



注：OCR はシステム・リセット時にクリアされる（Reset SW 押下時，Power ON 時）

ソフトウェア的にはSASIと同等に扱えるようになっています．合計80Mバイトの容量制限はありませんが，ドライブ数は2台までです．

◆ SASIインターフェース(PC9801-27)

▶ I/O ポートアドレス
80H，82H
▶割り込みレベル
$INT_3(IR_{91})$
▶ DMA チャンネル
#0

SASIインターフェースは二つのI/Oアドレスと，四つのレジスタから構成されています．インターフェースには特別な制御用シーケンスを持ったLSIは使用しておらず，基本的にはソフトウェアによって，HDDの制御シーケンス信号を作ります．また，データの受け渡しは，98内蔵のDMAにより転送されます．

図2-94にアドレス一覧を示します．

▶ ODR/IDR(80H/ライト・リード)

HDDに対して送る制御用コマンド&データの受け渡しをします．データの受け渡しをDMAでなくCPUが直接行う場合は，このポートを使用します．

▶ OCR(82H/ライト)

SASIインターフェースの制御用ポートです．

OCRの内容を図2-95に示します．

D_0：INTE……INTE=1のときに割り込みの発生が許可されます．

D_1：DMAE……DMAE=1でDMAモードになります．

D_3：RST……RSTを1→0に変化させたときに，HDDに対して「RST」信号を送ります．

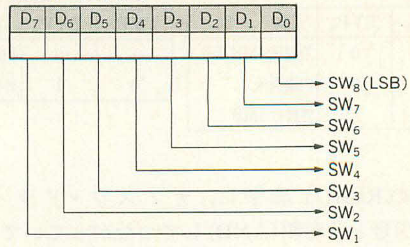
HDDのコントローラ(HDC)をリセットします．

D_5：SEL……SEL=1のときにHDDの制御信号である「SEL」をONにします．

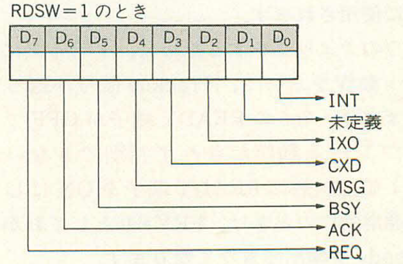
コントローラを選択するときONにします．コントローラ番号(01H)はデータ信号(ODR)から送ります．

〈図 2-96〉ISR

RDSW=0のとき



RDSW=1のとき



D_6：NRDSW……ISRレジスタで読み取れる内容を切り替えます．NRDSW=0でHDDインターフェース・ボード上のディップ・スイッチを読み出します．NRDSW=1でHDDからの制御信号を読み出します．

D_7：CHEN……CHEN=1で，インターフェースのデータ・制御線(内部バス)の出力が許可されます．内蔵型インターフェースでは無効です．

▶ ISR(82H/リード)

図2-96にISRを示します．

D_0：INT……割り込み線の状態を読み出します．

INTE=0の場合はリセット状態にされ，INT=1で割り込み状態となります．

D_2：IXO……HDD制御用の入力で，内部バスのIXOの状態を示します．

データ転送の方向(0=98→HDD/1=HDD→98)

D_3：CXD……HDD制御用の入力で，内部バスのCXDの状態を示します．

データ転送の情報(0=データ情報/1=制御情報)

D_4：MSG……HDD制御用の入力で，内部バスのMSGの状態を示します．

動作完了状態で，ポスト・ステータス・バイトが送られている(MSG=1)

D_5：BSY……HDD制御用の入力で，内部バスのBSYの状態を示します．

内部バスが動作中(BSY=1)

D_6：ACK……HDD制御用の入力で，内部バスのACKの状態を示します．

データ転送(1バイトごと)の応答信号で，IDR/ODRをアクセスしたときは“1”，REQ信号がオフに

〈図2-97〉WD33C93のI/Oアドレス

LSI	命令	I/Oポート・アドレス	R/W	データ D7	D6	D5	D4	D3	D2	D1	D0	備考
WD33C93	REGISTER ADDRESS SER	CC0	W	AR_7	AR_6	AR_5	AR_4	AR_3	AR_2	AR_1	AR_0	CONTROL REGISTERのアドレス・セット
	AUXILIARY STATUS	CC0	R	INT	LCI	BSY	CIP	0	0	PE	DBR	補助ステータス
	CONTROL REGISTER	CC2	R/W	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	WD33C93のコントロール・レジスタ
制御用レジスタ	STATUS READ	CC4	R	—	TCI	—	—	—	—	DMA_1	DMA_0	DMAチャネル・リード. TC割り込みのモニタ
	COMMAND WRITE	CC4	W	0	0	0	TCIR	TCMR	TCMS	DMER	DMES	DMAイネーブル. TCマスク・セット/リセット. TC割り込みリセット

なると“0”になります．

D_7：REQ……HDD制御用の入力で，内部バスのREQの状態を示します．

データ転送の要求信号(REQ＝1)

◈ SCSIインターフェース(PC9801-55)

▶使用LSI

WD33C93/A

▶ I/Oポート・アドレス

CC0H，CC2H，CC4H(標準設定)

▶割り込みレベル

INT_3(IR_{91})(標準設定)

▶ DMAチャネル

#0(標準設定)

SCSIインターフェースには多くの種類がありますが，もっともスタンダードなPC9801-55ボードについて解説します．これは，SCSIコントローラに，WD33C93/WD33C93A(以下33C93と略す)を使用し，転送プロトコルの一部をインテリジェントに処理するLSIです．

図2-97のように，I/Oポート・アドレスは，標準ではCC0H，CC2H，CC4Hを使用しますが，これは基板上のジャンパで，CD0H～CD4H，CE0H～CE4H，CF0H～CF4Hへ変更ができます．通常はCC0H～CC4Hで使用します．

割り込みレベルについても，INT_3以外に変更可能です．SASIインターフェース等と共存する場合に，INT_3以外に設定します．DMAチャネルも#0以外に変更可能ですが，拡張スロットには#1は出力されていませんし，80286以上のCPUを持つ機種の拡張スロットには#2も出力されていませんので，選択の幅は多くありません．

他にも設定として，ローカル・メモリ・アドレスがあります．機種，CPUに合わせて変更するディップ・スイッチで，使用する拡張ROMアドレスが変わります．

I/Oアドレスは三つ割り当てられていますが，このうち二つは33C93が使用します．アドレスは4種類に可変できますが，ここではCC0H～CC4Hを例に挙げて解説します．

▶ CC4H：DMAチャネル・リード，TC割り込みモニタ/リード

D_{0-1}：DMA_0-DMA_1……ボードに搭載されているDMAチャネルの切り替えのディップ・スイッチを読み出します．この設定に合わせて，使用するDMAチャネルを決定します．

D_6：TCI……$DMATC_0$信号により割り込みのモニタをします．TCI＝1で割り込み発生

▶ CC4H：DMAイネーブル，TCマスク，TC割り込み/ライト

D_0：DMES……DMAのイネーブル・セット

D_1：DMER……DMAのイネーブル・リセット

D_2：TCMS……TC割り込みマスクのセット

D_3：TCMR……TC割り込みマスクのリセット

D_4：TCIR……TC割り込みのリセット

● WD33C93

WD33C93は，SCSIコントローラというより，ディスク・コントローラの性格が強く，ヘッド，シリンダ，セクタ等の管理がLSI単体でできます．また，コンビネーション・コマンドを持ち，割り込み回数を削減できます．

▶ CC0H：33C93のレジスタ・アドレス・セット/ライト

アクセスする33C93のレジスタのアドレス番号を

〈図2-98(a)〉 WD33C93 レジスタ構成

アドレス	R/W	レジスタ	ビット D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	アドレス (16進)
CC0	W	Address	アドレス								—
	R	Auxiliary Status	INT	LCI	BSY	CIP	0	0	PE	DBR	—
CC2	R/W	Own ID	0	0	0	0	0	ID_2	ID_1	ID_0	00
	R/W	Control	DMA	WDB	0	0	0	0	HA	HPE	01
	R/W	Timeout Period	タイム・アウト定数								02
	R/W	Total Sectors	セクタ数/トラック								03
	R/W	Total Heads	ヘッド数								04
	R/W	Total Cylinders(MSB)	シリンダ数(上位バイト)								05
	R/W	Total Cylinders(LSB)	シリンダ数(下位バイト)								06
	R/W	Logical Address(MSB)	論理セクタ(最上位バイト)								07
	R/W	Logical address	論理セクタ								08
	R/W	Logical adress	論理セクタ								09
	R/W	Logical Address(LSB)	論理セクタ(最下位バイト)								0A
	R/W	Sector Number	セクタ番号								0B
	R/W	Head Number	ヘッド番号								0C
	R/W	Cylinder Number(MSB)	シリンダ番号(上位バイト)								0D
	R/W	Cylinder Number(LSB)	シリンダ番号(下位バイト)								0E
	R/W	Target LUN	ターゲット論理ユニット番号								0F
	R/W	Command Phase	0	コンビネーション・コマンドの終了状態							10
	R/W	Synchronous Transfer	0	転送サイクル TP_2	TP_1	TP_0	0	オフセット OF_2	OF_1	OF_0	11
	R/W	Transfer Count(MSB)	転送バイト数(最上位ビット)								12
	R/W	Transfer Count	転送バイト数								13
	R/W	Transfer Count(LSB)	転送バイト数 (最下位ビット)								14
	R/W	Destination ID	0	0	0	0	0	DI_2	DI_1	DI_0	15
	R/W	Source ID	ER	ES	0	0	SIV	SI_2	SI_1	SI_0	16
	R	SCSI Status	SCSI ステータス								17
	R/W	Command	SBT	コマンド・コード							18
	R/W	Data	入出力データ								19
	R/W	Momory Bank	0	ROM_0	0	0	MEM_1	IRE_1	WRS_1	0	30
	R	Memory Window	—	HST_1	HST_0	WND_4	WND_3	WND_2	WND_1	WND_0	31
	R/W	NEC リザーブ	使用禁止								32
	W	NEC リザーブ	使用禁止								33
	R	RESENT/INT	RRST	—	ILV_2	ILV_1	ILV_0	ID_2	ID_1	ID_0	33
	R/W	NEC リザーブ	使用禁止								34
	W	NEC リザーブ	使用禁止								35

〈図 2-98(b)〉 WD33C93 の I/O コマンド・ビット

ビット名	R/W	0/1	機　能	コマンド名
WRS$_1$	W	0	SCSI インターフェース・ボードからの SCSI バス・リセットを解除する	Memory
		1	本ビット書き込み後 SCSI バス上 $\overline{\text{RST}}$ 信号か "L" アクティブになる	
	R	0☆	SCSI バスが SCSI インターフェース・ボード側からはリセットされていないことを示す	
		1	SCSI の $\overline{\text{RST}}$ 信号を，SCSI インターフェース・ボードへの I/O コマンドによりアクティブにした．SCSI バスの他のデバイスからリセットされても本ビットは 1 にならない(リセットし続ける)	
IRE$_1$	R/W	0☆	SCSI インターフェース・ボードからシステム側への割り込みがマスクされる．ただし，割り込み要因自体を本ビットでクリアすることはできない	
		1	SCSI インターフェース・ボードからシステム側への割り込みが許可される．本ビットが "0" の間に割り込み要因が発生したときは，これを "1" にしてマスクを解除しても割り込みは発生しない．しかし，マスク解除前に割り込み要因をクリアすればこの限りではない	
MEM$_1$	W	0	ローカル・メモリ(ROM)をシステム側からアクセスできないようにする	Memory Bank
		1	ローカル・メモリ(ROM)をシステム側からアクセスできるようにする	
	R	0	ローカル・メモリ(ROM)をシステム側からアクセスできない	
		1☆	ローカル・メモリ(ROM)をシステム側からアクセスできる	
ROM$_0$	R/W	0☆	ROM バンクの設定：下	
		1	ROM バンクの設定：上	
WND$_{4\sim0}$	R	—	ホスト CPU から見えるローカル・メモリ・アドレスを切り替えるスイッチのビットが読める．内容はスイッチ設定の項を参照	Memory Window
HST$_{1,0}$	R	—	本体タイプに合わせて設定したスイッチのビットが読める．内容はスイッチ設定の項を参照	
ID$_{2,1,0}$	R	—	SCSI バス上での SCSI インターフェース・ボードの ID 番号を設定するスイッチ・ビットが読める．内容はスイッチ設定の項を参照	Reset /INT
ILV$_{2,1,0}$	R	—	システム側に上げる割り込みのレベルを設定するためのスイッチが読み込める．内容はスイッチ設定の項を参照	
RRST	R	0☆	最後に本コマンドを読んでから，他 SCSI 装置からバス・リセットが行われていないことを示す	
		1	SCSI バスの上の $\overline{\text{RST}}$ 信号が 25μs 以上 "L" アクティブになったとき "1" になる．本コマンド読み込み後，本ビットは "0" になる．ただし，I/O コマンドにより自分自身でバス・ラインをリセットしても本ビットは "1" にはならない 注：SCSI インターフェースでは，パワーオンまたはシステム・リセット時には本ビットを "0" にしているが，同時に SCSI バス上の他の装置からリセットがかかっている可能性があるために，リセット後は必ずしも本ビットが "0" であるとはいえない	

☆はシステムリセット後の設定

書き込みます．ここで指定されたレジスタが CC2H で読み書きできます．

▶ CC0H：33C93 の補助ステータス/リード

D$_0$：DBR……有効なデータがある

D$_1$：PE……パリティ・エラー

D$_4$：CIP

D$_5$：BSY

D$_6$：LCI

D$_7$：INT……割り込み確認

▶ CC2H：33C93 のコントロール・レジスタ・リード/ライト

図 2-98 は 33C93 のレジスタの一覧です．PC9801-55 ボードでは，これ以外にも 30H〜33H が拡張されていて，SCSI ドライブに RESET 信号を送ったりできます．

図 2-99 は 33C93 のコマンド一覧です．コマンド・レジスタで使用します．

〈図 2-99〉WD33C93 のコマンド

コマンド・コード(16進)	コマンド
00	Reset
01	Abort
02	Assert ATN
03	Negate ATN
04	Disconnect
05	Reselect
06	Select With ATN
07	Select Without ATN
08	Select With ATN and Transfer
09	Select Without ATN and Transfer
0A	Reselect and Receive Data
0B	Reselect and Send Data
0C	Wait For Select and Receive
10	Receive Command
11	Receive Data
12	Receive Message Out
13	Receive Unspecified Info Out
14	Send Status
15	Send Data
16	Send Message In
17	Send Unspecified Info In
18	Translate Address
20	Transfer Info
21	Transfer Pad

◈ IDE インターフェース

▶ I/O ポート・アドレス
640H, 642H, 644H, 646H, 648H, 64AH, 64CH, 64EH, 74CH, 74EH
▶割り込みレベル
$INT_3(IR_{91})$
▶ DMA チャネル
なし

IDE インターフェースは，98NOTE や PC9821 以降のデスクトップで使用されています．ソフトウェア的には，SASI インターフェースと同様になっていて，SASI インターフェースと同時に使用できません．

IDE は本来，AT 互換機専用のドライブで，PC/AT の標準 HD インターフェースであった WD1003 のコントロール回路を HDD 内部に搭載したものです．IDE ドライブにアドレス・デコーダとバス・バッファを付けるだけで，バスに直結できるように設計されています．

WD1003 は ST506 規格の HDD 用コントローラで，2 台までの HDD を制御できます．PC98 シリーズ用の SASI インターフェースも，ST506 の HDD に，HD コントローラを付ける形で 2 台まで増設できますから，よく似た形になります．

〈図 2-100〉IDE の I/O アドレス

レジスタ	I/O ポート・アドレス	R/W	備　考
データ・レジスタ	640-641	R/W	16 ビット
エラー・レジスタ	642	R	
ライト・プリコンペンセーション・レジスタ	642	W	
セクタ・カウント・レジスタ	644	R/W	
セクタ・ナンバ・レジスタ	646	R/W	
シリンダ・ロウ・レジスタ	648	R/W	
シリンダ・ハイ・レジスタ	64A	R/W	
ドライブ/ヘッド・レジスタ	64C	R/W	
ステータス・レジスタ	64E	R	
コマンド・レジスタ	64E	W	
オルタネート・ステータス・レジスタ	74C	R	
ディジタル・アウトプット・レジスタ	74C	W	
ディジタル・インプット・レジスタ	74E	W	

PC98 シリーズでの IDE インターフェースは，BIOS 未対応や増設オプションがない理由で，1 台しか接続できない機種や，使用できる HDD の容量の上限が固定されてしまっている機種もあるようです．PC9801BA2/BX2/PC9821Ap2/As2 等では，500MB の容量を持つドライブや，2 台までの接続を正規にサポートしています．

IDE インターフェースは，16 ビット幅のデータ・バスでデータ転送しますので，内蔵 DMA を使用せずに CPU がソフトウェアでデータ転送します．そのため，高速な CPU を持つ機種では内蔵 DMA よりも高速に転送ができます．

◈ IDE レジスタ

IDE の I/O アドレス・ポートは図 2-100 に示すように，16 ビット・バス×1，8 ビット・バス×9 の 10 個あります．データの転送は 16 ビット・バスで行われます．

▶ 640H(R/W)データ・レジスタ

データ・レジスタは 16 ビット幅です．コマンドにより次の 5 種類のデータ転送を行います．

(1) READ/WRITE…セクタ単位の転送
(2) READ/WRITE LONG…ECC のデータ転送
(3) READ/WRITE BUFFER…ドライブ内のデータ・バッファとの転送
(4) FORMAT…フォーマット用インタリーブ・テーブルの転送
(5) IDENTIFY…ID 情報等のデータ転送

▶ 642H(R)エラー・レジスタ

〈図 2-101〉エラー・レジスタの各ビット

ビット	名　称	意　味
7	Bad Block Detected	アクセス・セクタ ID に Bad Block マークが検出された(出荷時には Bad Block マークの書き込まれたセクタは存在しない．Format コマンドにより書き込まれる)
6	Data ECC Error	リード・コマンド実行時，データ領域に ECC により回復不能なエラーが発生した
5	—	(未使用)
4	ID Not Found	アクセス要求されたセクタが発見できない．リトライ・モードがイネーブルされている場合は，所定のリトライ動作を行う
3	—	(未使用)
2	Aborted Command	コマンドが実行途中で中断された．原因は，ステータス・レジスタに示される(Not Ready, Write Fault など)．無効コマンドの場合もセットされる
1	Trak0 Error	Restore コマンド実行時にトラック・ゼロが検出されなかった
0	DAM Error	リード・コマンド実行時，データ部のアドレス・マークが検出されなかった

〈図 2-102〉診断時のエラー・レジスタの値

値	意　味
01	ノー・エラー
02	コントローラ・レジスタ・エラー
03	バッファ RAM エラー
04	ECC 回路エラー
05	CPU ROM/RAM エラー
06·7F	(予約)

〈図 2-103〉ライト・プリコンペンセーション・レジスタの値

値	意　味
44H	Read/write Long コマンド時の ECC 長 7 バイト
55H	Read Ahead Cache オフ
AAH	Read Ahead Cache オン
BBH	Read/write Long コマンド時の ECC 長 4 バイト
その他	無効(Aborted Command Error 発生)

〈図 2-104〉シリンダ番号のフォーマット

シリンダ・ハイ・レジスタ

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
—	—	—	—	—	—	上位2ビット	

シリンダ・ロー・レジスタ

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
下位8ビット							

シリンダ番号

〈図 2-105〉ドライブ/ヘッド・レジスタ

ビット	名 称	意　味
7	—	ドライブ予約(1 に設定すること)
6	—	ドライブ予約(0 に設定すること)
5	—	ドライブ予約(1 に設定すること)
4	Drive Select	マスタ/スレーブ・モード時 0：マスタ・ドライブ 1：スレーブ・ドライブ シングル・モード時 0：ドライブ 1：ドライブは選択されず，レジスタに 00h が返送される
3〜0	Head Select	アクセスの先頭ヘッド番号を設定する．ヘッド番号は 0 から始まる

ステータス・レジスタの ERROR がセットされているときは，最後に実行されたコマンドのエラー情報を保持します．パワーオン直後の自己診断や，DIAG NOSE コマンド実行時の結果報告にも使用されます(図 2-101，図 2-102)．

▶ 642H(W)ライト・プリコンペンセーション・レジスタ

設定値と機能を図 2-103 に示します．

▶ 644H(R/W)セクタ・カウント・レジスタ

3 種類のコマンドの実行時にパラメータ/返り値の設定用です．

(1) READ/WRITE/READ VERIFY…転送されるセクタ数

(2) SET PARAMETERS…1 トラック当たりのセクタ数の指定

(3) POWER CONTROL…動作モードに対応する返り値が設定される

▶ 646H(R/W)セクタ・ナンバ・レジスタ

ディスク・アクセス・コマンド(READ/WRITE/READ VERIFY)実行時に，アクセスする先頭セクタ番号を指定します．

▶ 648H(R/W)シリンダ・ロウ・レジスタ

アクセスする先頭シリンダ番号の下位ビットを指定します．

シリンダ番号のフォーマットを図 2-104 に示します．

▶ 64AH(R/W)シリンダ・ハイ・レジスタ

アクセスする先頭シリンダ番号の上位ビットを指定します．

▶ 64CH(R/W)ドライブ/ヘッド・レジスタ

アクセスするドライブ番号，ヘッド番号を指定します(図 2-105)．

▶ 64EH(R)ステータス・レジスタ・レジスタ

このレジスタを読むとインタラプトがクリアされます(図 2-106)

▶ 64EH(W)コマンド・レジスタ

ドライブに実行させるコマンドを書き込みます．コマンドは 13 種類ですが，ステータス・レジスタのビットが以下の状態のときのみコマンドを受け付けます．

〈図 2-106〉ステータス・レジスタ

ビット	名　称	意　味
D_7	BUSY	次の場合に"1"にセットされる． (1)ホスト・システムからRESET信号によりハード・リセットされた場合，またはFixed Diskレジスタの RESETビットによりソフトウェア・リセットされた場合 (2)ホスト・システムからコマンドを書き込まれてからコマンド処理が終了するまで．ただし，データ転送要求中(Data Requestビットが"1"にセットされる)はリセットされる
D_6	READY	本ビットが"1"で，かつSEEK COMPLETEビットが"1"のとき(Seekコマンド直後のみ例外)，ドライブはレディ状態にあり，ホスト・システムからのコマンドを受け付ける． "0"のとき，ドライブはレディ状態(Ready)になく，ホスト・システムからのコマンドを受け付けない．コマンド実行中にノット・レディ(Not Ready)状態が発生した場合は，コマンドは中断され，次のコマンド書き込みまではドライブ状態(Ready/Not Ready)にかかわらず，リセット状態が保持される．このとき，ERRORビット(ビット0)もセットされる パワーオン直後はリセットされ，ドライブが定常回転になり，コマンド受信可能状態になった時点でセットされる． スタンバイ状態で，スタンバイ解除コマンドを受信したときもリセットされ，コマンド受信可能状態になった時点でセットされる
D_5	WRITE FAULT	"1"はデータ書き込みコマンド実行時，書き込み異常が発生し，データが正しく書き込まれなかったことを示す．このときにERRORビット(ビット0)もセットされ，コマンドは中断される．次のコマンド発行までは，Write Fault状態の有無にかかわらず，セット状態が保持される．ホスト・システムから次に発行されたコマンドによりリセットされる
D_4	SEEK COMPLETE	シーク動作をともなうコマンド実行時，シーク動作が正常に終了しなかった場合に"0"にリセットされる．このときERRORビット(ビット0)もセットされ，コマンドは中断される．つぎのステータス・レジスタ読み取りまでは，リセット状態が保持される．ステータス・レジスタ読み取りが生じると，その時点でのSEEK COMPLETE状態を表示する．シーク・コマンドはシーク動作の終了を待たずにコマンドを正常終了するので，コマンド終了時には本ビットはセットされない．このとき，ドライブはレディ状態にあり，コマンドを受け付ける．パワーオン直後はリセットされ，ドライブが定常回転になり，コマンド受信可能状態になった時点でセットされる． スタンバイ状態で，スタンバイ解除コマンドを受信したときもリセットされ，コマンド受信可能状態になった時点でセットされる
D_3	DATA REQUEST	データ転送をともなうコマンド実行時，ドライブがデータ転送準備ができた場合に"1"にセットされる
D_2	CORRECTED DATA	"1"は，データ・リード時に読み取りエラーが発生したが，ECCにより訂正されたことを示す．マルチセクタ・リード動作は中止されない
D_1	INDEX	ドライブの1回転に一度，出力されるパルス信号
D_0	ERROR	ホスト・システムから発行されたコマンド処理中に，何らかのエラーが発生したことを示す．エラー原因は，本ステータス・レジスタまたはエラー・レジスタに示される．ホスト・システムから次に発行されたコマンドによって，リセットされる(ただし，次に発行されるコマンドがFormatコマンド，Write Sectorsコマンドの場合，コマンドの書き込み時にはリセットされず，最初のインタラプト発生までの間にリセットされる)．マルチセクタ処理コマンドは中断される

(1) WRITE FAULT＝0

(2) READY＝1

(3) SEEK COMPLETE＝1

このレジスタを読むと，インタラプトがリセットされます(図2-107)．

▶ 74CH(R)オルタネート・ステータス・レジスタ

「ステータス・レジスタ・レジスタ」と同様ですが，インタラプトはクリアされません．

▶ 74CH(W)ディジタル・アウトプット・レジスタ

ディジタル・アウトプット・レジスタ・コマンドは図2-108のとおりです．

▶ 74EH(W)ディジタル・インプット・レジスタ

ディジタル・インプット・レジスタ・コマンドを図2-109に示します．

〈図 2-107〉IDE ドライブ・コマンド

コマンド名	コマンド・コード								使用されるパラメータ・レジスタ					
	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	DR	CY	HD	SN	SC	WP
Restore	0	0	0	1	X	X	X	X	Y	N	N	N	N	N
Read Sector(s)	0	0	1	0	0	0	L	R	Y	Y	Y	Y	Y	N
Write Sector(s)	0	0	1	1	0	0	L	R	Y	Y	Y	Y	Y	N
Read Verify	0	1	0	0	0	0	0	R	Y	Y	Y	Y	Y	N
Format Track	0	1	0	1	0	0	0	0	Y	Y	Y	N	Y	N
Seek	0	1	1	1	X	X	X	X	Y	Y	Y	Y	N	N
Diagnose	1	0	0	1	0	0	0	0	Y	N	N	N	N	N
Set Parameters	1	0	0	1	0	0	0	1	Y	N	Y	N	Y	N
Power Control	1	1	1	0	0	X	X	X	Y	N	N	N	Y	N
Read Sector Buffer	1	1	1	0	0	1	0	0	Y	N	N	N	N	N
Write Sector Buffer	1	1	1	0	1	0	0	0	Y	N	N	N	N	N
Identify Drive	1	1	1	0	1	1	0	0	Y	N	N	N	N	N
Set Features	1	1	1	0	1	1	1	1	Y	N	N	N	N	Y

レジスタ

DR：ドライブ/ヘッド・レジスタの Drive Select ビット
CY：シリンダ・ハイ/ロウ・レジスタ
HD：ドライブ/ヘッド・レジスタの Head Select ビット
SN：セクタ・ナンバ・レジスタ
SC：セクタ・カウント・レジスタ
WP：ライト・プリコンペンセーション・レジスタ
Y：コマンド発行に先だちパラメータ設定が必要
N：コマンド発行に先だちパラメータ設定が不要

コマンド・モード

X：Don't care
L："1" のとき Long コマンド(ECC バイト転送)，"0" のときノーマル・リード/ライト
R："0" のときリトライ実行，"1" のときリトライ禁止

マウス・インターフェース

▶使用 LSI
8255A
▶ I/O アドレス
7FD9H，7FDBH，7FDDH，7FDFH(8255)
BFDBH(タイマ)
▶使用割り込み
IR_6(変更可能な機種もある)
▶初期設定命令
8255＝93H

マウスとのインターフェースにはプログラマブル・パラレル・ポートの μPD8255A の互換品(以降，8255 と略す)を使用しています．このパラレル・ポートを介して，マウス・コントローラ IC を操作してマウスの移動量と，マウスのトリガ・スイッチが読み出せます．また，マウス用タイマ IC があり，通常は約 8 ms ごとに割り込みをかけて，マウスの移動量を監視して，マウス・ドライバ内での位置情報やトリガ・スイッチの状態情報を更新し，アイコンでの表示の必要性があれば表示を行います．

タイマからの割り込みは，8.3 ms～66.7 ms までの 4 種類に変更できるようになっています．割り込み時間の設定は，8255 とは別アドレスの BFDBH で行います．

使用されている 8255 は，マウス関連情報以外にも，ディップ・スイッチの内容やシステム情報が見えるようになっています．ただし，機種によって内容が変わります．

一般に 98 用のマウスは 2 ボタン式(右・左)ですが，EPSON-PC シリーズや，比較的新しい NEC-98 シリーズ等，機種によっては本体側は 3 ボタン対応しています．ただ，98 用 3 ボタン・マウスはあまり市販されてはいないようです．

マウス・インターフェースの 8255 は，ポート A/B/C(下位)がモード 0 で入力に，ポート C(上位)がモード 0 で出力に初期設定されています．モード・セット・コマンドは 93H です．

◈ マウスのしくみ

マウスには，X 軸，Y 軸方向の移動に合わせて，パルスが発生するスイッチが付いています．このパル

〈図 2-108〉ディジタル・アウトプット・レジスタ

ビット	名　称	意　味
$D_{7\sim4}$	—	(未使用)
D_3	—	ドライブ予約(1に設定すること)
D_2	Reset	ホスト・システムによるソフト・リセットとして機能する．"1" の間ドライブはリセット状態となる．このとき内部レジスタではすべてリセットされ，ステータス・レジスタのBUSYビットがセットされる．マスタ/スレーブ・モード時は，ドライブ/ヘッド・レジスタのDrive Selectビットにかかわらず，両ドライブともリセットされる
D_1	IEN	"0(アクティブ)" のとき，Drive Selectビット(ドライブ/ヘッド・レジスタ)により選択されたドライブからのホスト・インタラプト信号IRQをイネーブルする．"1" のとき，未処理インタラプト(Pending interrupt)の有無にかかわらず，IRQ出力はハイ・インピーダンス状態になる
D_0		(未使用)

〈図 2-109〉ディジタル・インプット・レジスタ

ビット	名　称	意　味
D_7	—	(未使用．読み取り時ハイ・インピーダンス状態となる)
D_6	−WG	ライト・ゲート信号．ドライブがデータを媒体上に書き込み中にアクティブとなる
$D_{5\sim2}$	−Head Select	ドライブ/ヘッド・レジスタのHead Selectビットの1の補数を示す
D_1	−Drive Set 1	ドライブ/ヘッド・レジスタのDrive Selectビットが "1" のとき(Slave Drive選択)，アクティブ("0")となる
D_0	−Drive Set 0	ドライブ/ヘッド・レジスタのDrive Selectビットが "0" のとき(Singleモードまたは Master/Slaveモードで Master Drive選択)，アクティブ("0")となる

スを，マウス・コントローラICの内部の8ビットのカウンタで計測して，現在位置を計算します．移動量が256カウントを越えてしまうと，マウス・コントローラICでは，自分の位置を見失ってしまいます．

コンピュータのプログラム(マウス・ドライバ)は，マウスの移動が256カウント以内に，再度マウス・コントローラICからデータを取得して，その差を算出し絶対位置を計算します．このために，マウス・インターフェースでは，一定時間ごとに割り込み(インターバル・タイマ)を発生し，最新のデータ取得と計算をします．

マウス・インターフェースの8255は，マウス・コントローラICのデータを一度に4ビットずつしか読み出せません．このため，8ビットのカウンタを，上位4ビット/下位4ビットに分けで2回の操作で読み出し，X/Y軸で合計4回の操作を必要とします．

マウス・インターフェースの構成を図2-110に，I/Oアドレス一覧を図2-111に示します．

◈ ポートA(7FD9H/リード)

▶ D_0〜D_3：MD_{0-3}

マウス位置データで，SXY，SHLで指定されたデータが，バイナリ4ビットで出力されます．

▶ D_5：RIGHT

0で右トリガ・スイッチが押されています．

▶ D_6：CENTER

0で中央トリガ・スイッチが押されています．

▶ D_7：LEFT

0で左トリガ・スイッチが押されています．

◈ ポートB(7FDBH/リード)

▶ D_1(D_0〜D_1)：SPDSW(FSH，FSL)

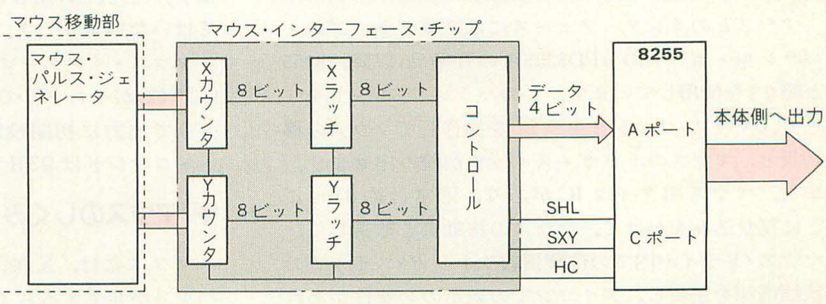
CPUクロック・スピードを示します．機種によって異なり使用されていないものもありますが，0で高速モードです．

▶ D_6：RAMKL

ディップ・スイッチSW_{3-6}の設定状態です．0で内部RAMの512KB〜640KBを切り離します．

EPSON-PCシリーズでは，ポートBを出力モードにして各種設定を行う機種もありますが，機種ごとに内容が違います．

〈図 2-110〉
カウンタICの内部ブロック図



〈図 2-111〉マウス・インターフェースの I/O アドレス

LSI	命　令	I/Oポート・アドレス	R/W	データ								備　考
				D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
μPD8252	ライト・モード	7FDF	W	1	0	0	1	0	0	1	1	モード・セット
	ライト・ポートC	7FDF	W	0	0	0	0	1	0	0	0/1	割り込み Enable 0：Enable　1：Disable
		7FDF	W	0	0	0	0	1	1	1	0/1	Clear Count(HC) 0：クリアしない　1：クリアする
	ライト・ポートC	7FDD	W	HC	SXY	SHL	$\overline{INT}$	0	0	0	0	ポートCはこの命令でも変更できる
	リード・ポートC（＊1）	7FDD	R	HC	SXY	SHL	$\overline{INT}$	MODSM	CPUSW	SW_{1-6}	SW_{1-5}	下位4ビットによりスイッチの状態を読み取る
	リード・ポートB（＊2）	7FDB	R	—	RAMKL	—	—	—	—	SPDSW	—	ディップ・スイッチを読み込む
	リード・ポートA	7FD9	R	LEFT	×	RIGHT	×	MD_3	MI_2	MD_1	MD_0	マウスの状態を読み取る
	ライト・タイマ（＊2）	BFDB	W	0	0	0	0	0	0	T_1	T_0	割り込みタイマを設定する

＊1：PC9801/E/F1,2,3/M2,3/U2/VF2/VM0,2,4/UV2 では，下位ビットは未定義
＊2：PC9801RX/RA/DX/DS/DA/CS,PC-98XL/XL2/RL/GS のノーマル・モードではダミー
PC9801/E/F1,2,3/M2,3/U2/VF2/VM0,2,4/UV では存在しない

◈ ポート C 下位(7FDDH/リード)

▶ D_0：SW_{1-5}

ディップ・スイッチ SW_{1-5}の設定状態と RS-232-C 同期モード設定状態を示します．

▶ D_1：SW_{1-6}

ディップ・スイッチ SW_{1-6}の設定状態と RS-232-C 同期モード設定状態を示します．

▶ D_2：CPUSW

ディップ・スイッチ SW_{3-8}の設定状態を示します．

0＝80286，386，486，Pentium
1＝70116，70136

▶ D_3：MODSW

ノーマル/ハイレゾ・モードの設定状態(ハイレゾ機種のみ)を示します．

0＝ハイレゾ・モード
1＝ノーマル・モード

◈ ポート C 上位(7FDDH/ライト)

▶ D_4：INT

0 でマウスのタイマ割り込みを許可します．

▶ D_5：SHL

MD_{0-3}に出力されるデータの切り替えで，0 のとき下位 4 ビット，1 のときに上位 4 ビットのデータを出力します(図 2-112)．

▶ D_6：SXY

MD_{0-3}に出力されるデータの切り替えで，0のときX軸方向，1のときにY軸方向のデータを出力します．

〈図 2-112〉D_5：SHL

SXY	SHL	データ
0	0	X 軸方向　下位 4 ビット・データ
0	1	X 軸方向　上位 4 ビット・データ
1	0	Y 軸方向　下位 4 ビット・データ
1	1	Y 軸方向　上位 4 ビット・データ

〈図 2-113〉マウス割り込み周期

値(16 進)	D_1	D_0	周波数	時　間
00	0	0	120Hz	8.3ms
01	0	1	60Hz	16.7ms
02	1	0	30Hz	33.3ms
03	1	1	15Hz	66.7ms

▶ D_7：HC

0 のときは，読み出した時点の移動データが MD_{0-3} に出力されます．

0 → 1 のときに MD_{0-3}のデータがラッチされ，この際カウンタはクリアされます．

1 のときはラッチされたデータが MD_{0-3}に出力されます．

◈ マウス割り込み間隔設定(BFDBH××・ライト)

マウス割り込み周期は図 2-113 のように D_1～D_0を設定することで選ぶことができます．

プリンタ・インターフェース

▶使用 LSI

μPD8255A 相当

▶ I/O アドレス

40H, 42H, 44H, 46H

▶使用割り込み

IR_8

▶初期設定命令

8255＝82H(他にも，システム・ポート用の 8255 のポート C・ビット 7 を使用している)

プリンタ・インターフェースには，μPD8255A の相当品(以降，8255 と略す)が使用されています．プリンタへのデータ出力(8 ビット)はポート A が，BUSY 信号入力はポート B・ビット 2 が，ストローブ信号出力はポート C・ビット 7 の合計 10 本の必要最小限の信号線だけになっています．

プリンタ用割り込みとしてポート C・ビット 3 が 8259(スレーブ)の IRQ_0に接続されていますが，ノーマル・モードでは ACK 信号入力(ポート C・ビット 6)が使用されていないために，8255 をモード 1 に設定できず活用されていません．しかし，ハイレゾ・モードではフル・セントロニクス仕様となっていて割り込みも活用されてます．

ノーマル・モードとハイレゾ・モードのプリンタ・インターフェースのハードウェアは異なります．

プリンタ・インターフェース用の 8255 ではプリンタ制御以外にも，ポート B からシステム情報を読み出すことができます．比較的機種依存性の高い情報が多くなっていますので，機種別にどのような信号が出ているか把握したうえで使用しなくてはなりません．

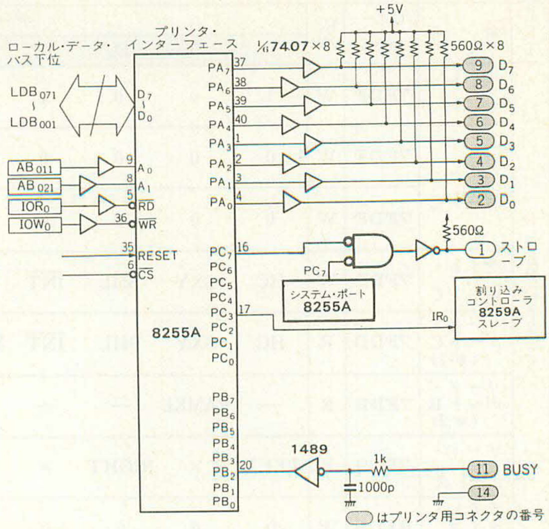
プリンタ・インターフェースの 8255 は，ポート A/C がモード 0 で出力に，ポート B がモード 0 で入力に初期設定されています．モード・セット・コマンドは 82H です．図 2-114 にインターフェース回路を，図 2-115 に I/O アドレス一覧を示します．

◈ ポート A(40H/ライト)

データ出力ポートで書き込んだデータは，プリンタ・インターフェース・コネクタ PDB_0～PDB_7へバッファを通して出力されます．書き込んだデータは 8255 でラッチされるので，前に書いたデータを読み出すこともできます．

8255 とプリンタ・インターフェース・コネクタの間にバッファが入っているために，ポート A で入力も設定しても，プリンタ・ポートを入力として使用することはできません．

〈図 2-114〉 プリンタ・インターフェース回路



◈ ポート B(42H/リード)

▶ D_0：VF

PC9801VF でのみ “1” になり，その他の機種では “0” になります．

▶ D_1：CPUT

実際に動作している CPU の種別を示します．EPSON-PC シリーズではディップ・スイッチ $SW_{3\text{-}8}$の設定がそのまま見えます．

0：80286/386/486/Pentium

1：70116

▶ D_2：BSY

プリンタ・コネクタの BUSY にインバータを通してつながっていて，インバータのために論理が反転しています．

0：BUSY

1：READY

▶ D_3：HGC

機能拡張状態を示します．HCG＝0 のとき拡張機能使用を示し，16 色表示機能の使用，後続描画機能の使用等を行っているかの状態表示を行います．

EPSON-PC シリーズはディップ・スイッチ $SW_{1\text{-}8}$の設定が見えます．

▶ D_4：LCD

プラズマ・ディスプレイ使用/未使用を示します．LCD＝0 でプラズマ・ディスプレイ使用です．

EPSON-PC シリーズはディップ・スイッチ $SW_{1\text{-}3}$の設定が見えます．

▶ D_5：MOD

CPU クロックの状態を示します．システム・クロッ

〈図 2-115〉 プリンタ I/O アドレス

μPD8255	命令	I/Oポート・アドレス	R/W	データ D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	備考
プリンタ・ポート・コントロール・レジスタ	ライト・モード	46	W	1	0	0	0	0	0	1	0	8255A モード・セット
	ライト・シグナル1	46	W	0	0	0	0	0	0	1	1/0	80287/80387(SX)のリセット制御 0：シャットダウン時リセットしない 1：リセットする (＊1)
		46	W	0	0	0	0	0	1	1	1/0	IR_8の ON/OFF 0：アクティブ 1：インアクティブ
		46	W	0	0	0	0	1	1	1	1/0	PSTB の ON/OFF 0：アクティブ 1：インアクティブ
ポートC	ライト・シグナル2	44	W	$\overline{PSTB}$	0	0	0	IR_8	0	RST287 (387)	0	PSTB，IR_8，RST 287/387 は本命令でも制御可能 (＊1)
ポートA	ライト・データ	40	W	WD_8	WD_7	WD_6	WD_5	WD_4	WD_3	WD_2	WD_1	プリンタにデータを送る
	リード・データ(診断用)	40	R	WD_8	WD_7	WD_6	WD_5	WD_4	WD_3	WD_2	WD_1	ライト・データでセットしたデータを読み込む
ポートB	リード・シグナル1 (＊2)	42	R	TYP_1	TYP_0	MOD	LCD	HGC	$\overline{BSY}$	CPUT	VF	プリンタの状態およびCPUのモード・タイプを読み込む
ポートC	リード・シグナル2	44	R	$\overline{PSTB}$	×	×	×	IR_8	×	RST287 (387)	×	8255A のポートCの状態を読み込む
システム・ポート・コントロール・レジスタ	ライト・ポートC (＊3)	37	W	0	0	0	0	1	1	0	1/0	PSTB 信号 Enable F/F の ON/OFF 0：アクティブ 1：インアクティブ

＊1：RST287/387 は 80286/386/486/Pentium 搭載機種のみ
＊2：PC9801/E/F1,2,3/M2,3 では，LCD，HGC，CPUT は未定義
　　VF は，PC9801VF2 でのみ 1，他はつねに 0，PC9801 では $\overline{BSY}$ のみ使用．PC9801 では MOD も未定義
＊3：PC9801 では，$\overline{PSTB}$ Enable F/F は，I/O ポート・アドレス 94H(IMB フロッピ・ディスク・インターフェースの外付けレジスタ)の D_4 ビットを使用
×：不定
IR_8：プリンタ制御回路から 8259 への割り込み信号

クの判別(5 MHz/8 MHz)ができ，8253(タイマ LSI)の入力クロックを知るために使用します．

0：5/10/12/20/25/40 MHz(5/10 MHz 系)

1：8/16/33/60/66 MHz(8 MHz 系)

CPU クロックとシステム・クロックとの関係は，機種によっては上記の内容と異なる場合がありますが，MOD で得られるデータは初代 PC9801 を除いて，8253 の入力クロックを表します．

初代 PC9801 は，プリンタ・コネクタの 13 ピンに接続されていますので，システム・クロックを知ることはできません．

▶ D_6〜D_7：TYP_0，TYP_1

システムのタイプを図 2-116 のように設定します．

〈図 2-116〉 システム・タイプの設定

TYP_1	TYP_0	システム・タイプ
0	0	PC9801
1	1	PC9801U2
0	1	未定義
1	0	上記以外の機種

◈ ポートC(モード0・44H/出力)

▶ D_1：RST287(RST387)

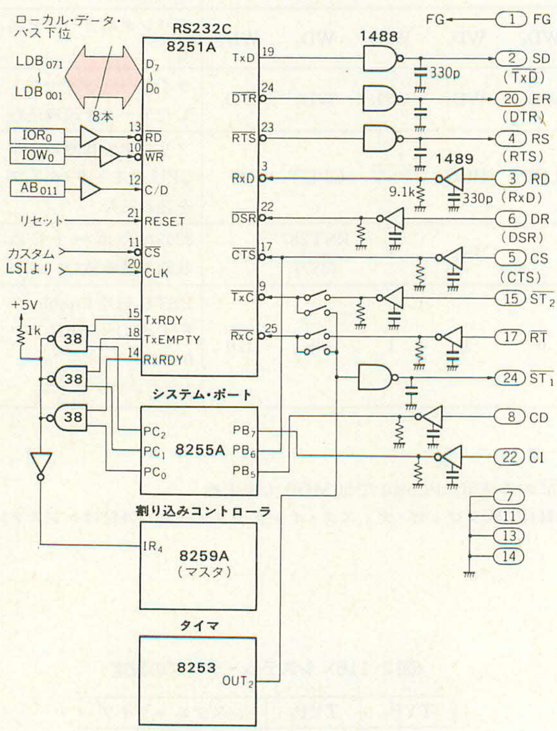
CPU リセット発生時，NDP または CPU 内蔵の NDP 機能をリセットするかどうかの指定です．このビットは 80286/386/486/Pentium CPU 動作時のみ意味を持ちます．

〈図 2-118〉 RS-232-C 関連 I/O アドレス

LSI	命令	I/Oポート・アドレス	R/W	データ D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	備考
8251	モード(A)	32	W	S_2	S_1	EP	PEN	L_2	L_1	B_2	B_1	μPD8251 動作モードの設定(非同期)
	モード(B)	32	W	SCS	ESD	EP	PEN	L_2	L_1	0	0	μPD8251 動作モードの設定(同期)
	コマンド	32	W	EH	IR	RS	RST	SBR	REN	ER	TEN	
	ステータス	32	R	DR	SYN	FE	OE	PE	TE	RRDY	TRDY	
	データ・リード	30	R	RD_8	RD_7	RD_6	RD_5	RD_4	RD_3	RD_2	RD_1	
	データ・ライト	30	W	SD_8	SD_7	SD_6	SD_5	SD_4	SD_3	SD_2	SD_1	
8253	カウンタ・セット	75	W	C_7 C_{15}	C_6 C_{14}	C_5 C_{13}	C_4 C_{12}	C_3 C_{11}	C_2 C_{10}	C_1 C_9	C_0 C_8	
	カウンタ・モード	77	W	SC_1	SC_0	RL_1	RL_0	M_2	M_1	M_0	BCD	
8255	マスク・セット	35	W	×	×	×	×	×	TXRE	TXEE	RXRE	
	リード・シグナル	33	R	CI	CS	CD	×	×	×	×	×	$\overline{CI}$ は PC 9801 では無効

×印：不定

〈図 2-117〉 RS-232-C インターフェース



0：リセットしない

1：リセットする

▶ D_3：IR_8

プリンタ用割り込みです．8259(スレーブ)の IRQ_1 に接続されていますが，前述のとおり，ノーマル・モードでは利用されていません．

0：ON

1：OFF

CPU が 80286/386/486/Pentium で動作している場合は，IR_8は NDP が使用します．

〈図 2-119〉 8251 に必要なクロック

動作条件	8251 入力クロック	最大遅延時間 (28 クロック)
システム・クロック 8MHz	1.9968MHz	14.03μs
システム・クロック 5/10MHz	2.4576MHz	11.40μs
PC9821Af, Ne	9.8304MHz	2.85μs

注意：ステータスの更新はステータスに影響を与える事象が起こってから最大 28 クロック(μPD8251 の入力クロック)周期の遅延がある．したがってステータス更新までに必要な最大遅延時間は上のようになる．I/O ポートを参照する場合は注意すること

▶ D_7：PSTB

プリンタ・ストローブ信号の出力です．システム・ポートの PC_6と AND を取って，プリンタ・コネクタの PSTB に接続されています．プリンタ・ポートやシステム・ポートの 8255 を初期化する場合は，その設定順序に注意が必要です(詳細はシステム・ポートの章参照)．

0：LOW

1：HIGH

RS-232-C インターフェース

▶使用 LSI

μPD8251A 相当

▶ I/O アドレス

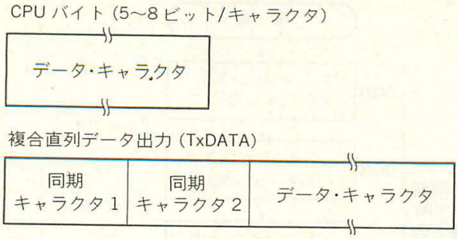
30H，32H

▶使用割り込み

IR_4(ベクタ# 0CH)

98 シリーズの RS-232-C インターフェースには，

〈図 2-120〉同期モードでの送信フォーマット



μPD8251A の互換品(以下，8251 と略す)を使用しています．その他にも，ボーレート生成用にタイマ LSI (8253) と割り込みマスク，外部信号読み出し用にパラレル・ポート(8255)を使用しています．

図 2-117 にインターフェース回路を，図 2-118 に関連 I/O アドレス一覧を示します．

◈ 8251 について

8251 は，同期/調歩同期(非同期)モードを持ちます．同期モードでは，同期キャラクタ数が 1～2，内部/外部同期検出，自動同期キャラクタ挿入等が可能になります(98 シリーズでは外部同期検出に必要な SYNC 端子が RS-232-C コネクタに接続されていないので，外部同期検出モードは使えない)．

8251 に必要なクロックには 2 種類あって，TxC/RxC は RS-232-C の通信ボーレートを決定するために，8253 で発生されたクロックが入力されています．

もう一つの CLK は，LSI 内部の動作タイミングを作るもので，LSI の処理速度に影響を与えますが，ボーレート等とは無関係です．クロックには，1.9968 MHz か，2.4576 MHz，9.8304 MHz が入力されています(図 2-119)．

8251 の仕様では，CLK のクロックは，TxC/RxC のクロックの 4.5 倍(同期モード時は 30 倍)である必要があります．例えば，CLK に 2.4576 MHz を使用している場合は，計算上では 34133 bps 以上出せないことになりますので，実質的には 19200 bps が上限になってしまいます．しかし，実際には 38400 bps で使用しても問題はなさそうです．

調歩同期モードではボーレート設定を，×1，×16，×64 の 3 種類から選べます．これは，TxC/RxC から入力されたクロックから分周して通信ボーレートを作るときの分周比を指定するもので，一般的には×16 を使用します．×1 を使用した場合は，同期通信と同様に TxC/RxC に相手側の 8251 と同一位相のクロックを与える必要があります．

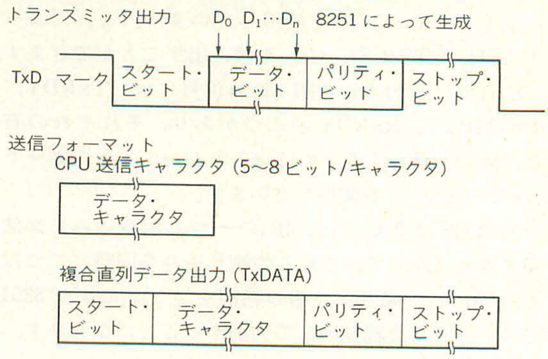
●ボーレート・ファクタの特徴

×1　最も高速な通信が可能．
　　ただし，送受信で相手と CLK 同期を取る必要がある．

×16　同期を取る必要が特になく，一般的な速度をサポート．

×64　同期を取る必要が特になく，内部分周比が大きいので低速通信向け．

〈図 2-121〉調歩同期モードでの送信フォーマット



◈ 同期モードと調歩同期モード

▶同期モード

送信側が書き込んだデータをそのままシリアルに変換して受信側に送る方式で，各キャラクタの先頭ビットを検出する(同期を取る)必要があります．同期を取る方法としては，以下の二つが挙げられます．一つは特定ビット列(同期キャラクタ)のサンプリングで同期を取る方法です．

もう一つは，端子入力によって同期を取る方法があります．

同期モードは，一般にはあまり使われていないモードで，専用モデム等に使われます．

同期モードでの送信フォーマットを図 2-120 に示します．

▶調歩同期モード

送信側が書き込んだデータの単位ごと(5～8 ビット)に特定の信号を付加して，受信側に送ります．その特定の信号には，スタート・ビットとストップ・ビットとパリティ・ビットがあります．スタート・ビットは各キャラクタの先頭ビットの前に，ストップ・ビットは各キャラクタの後尾ビットの後に付加します．パリティ・ビットを付加する場合は，後尾ビットとストップ・ビットの間に挿入されます．

これらの付加されたビットのために，同期モードに比べて転送効率が悪くなりますが，送信側と受信側の双方の同期を取る必要がないので，手軽に使用でき，一般的に使われています．汎用の RS-232-C は，ほとんどこの調歩同期方式になっています．

調歩同期モードでの送信フォーマットを図 2-121 に示します．

◈ 8255(システム・ポート)

8251 単体だけでは読み取ることができない外部信

号を読み出すために，システム・ポート用の8255のポートB・ビット5〜7を使用しています．これによって，CD，CTS(CS)，CI，を読み出すことができます．

8251から出力される割り込み信号には，TxRDY，TxEMPTY，RxRDYの三つがあり，それぞれの有効，無効を設定するためにシステム・ポートのポートC・ビット0〜2を使用しています．

内蔵RS-232-Cでは，IR_4の一つの割り込みしか使用できませんので，これらの割り込みを同時に二つ以上有効にした場合は，その判別をソフトにより8251のステータスを読み出して判別することになります．

◈ 8253(タイマLSI)とボーレート

RS-232-Cインターフェースのボーレートは，タイマLSIの8253で生成されます．使用しているのは#2チャネルで，モード3(方形波レート・ジェネレータ)で使われています．8253に入力されるクロックは機種によって2種類あり，これを認識して8253の分周比を決めないとボーレートが合わなくなるので注意が必要です．

8253に入力されるクロックは，CPUクロックが8MHz系のときは1.9968MHz，5MHz系のときには2.4576MHzになります．これを調べるには，プリンタ・ポート用の8255の42Hのビット5を読み取り，1ならば1.9968MHz，0ならば2.4576MHzであることがわかります．

RS-232-Cでは，普通，1200，2400，4800……等と倍々のボーレートを使用します．8251で調歩同期式(非同期)を使う場合は，ボーレートの16倍のクロックが必要になるので，19200bpsでは，307.2kHzが必要な計算になり，5MHz系の場合は1/8に分周すれば良いのですが，8MHz系の場合は1/6.5となり，8253では分周できません．この理由から，8MHz系クロックを持つ機種ではRS-232-Cのボーレートの上限が9600bpsに限定されてしまいます．

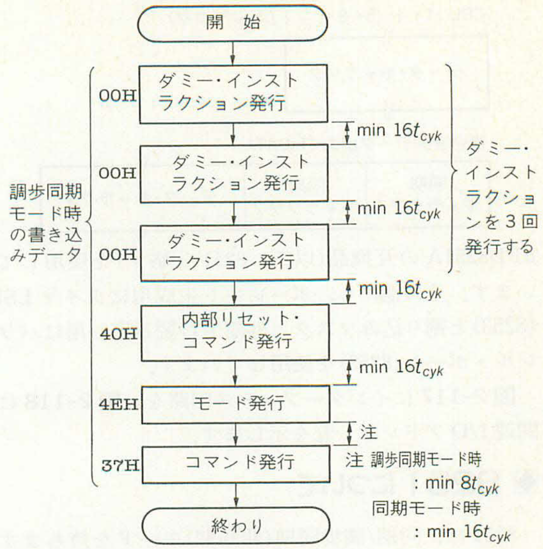
◈ 8251のレジスタ

8251のI/Oポートは二つあり，「モード・コマンド書き込み(設定)/ステータス読み出しポート」と，「データの読み書きポート」です．

モード・コマンド設定ポートは一つのアドレスを共用しており，モード設定(初期化データ5EHを書き込む)は8251をリセットした直後に一度だけ設定でき，二度目からはコマンド設定ポートになります．8251のリセットはコマンド設定(ソフトウェア)から行うので，図2-122の手順でリセット→モード設定する必要があります．

8251の各レジスタの内容は以下のとおりです．

〈図2-122〉8251のリセット



◈ モード設定(32H/ライト)

モード設定には，同期モード，調歩同期モードの2種類があり，D_6〜D_7のビットの意味が変わります(図2-123)．

▶ D_0〜D_1：B_0〜B_1

送受信のボーレートと，TxCLK，RxCLKの関係を規定します．ボーレートに対して送受信クロックの周波数が1倍か，16倍か，64倍かを選択します．

同期モードでは，$B_0=0$，$B_1=0$にします．

・D_2〜D_3：L_0〜L_1

1キャラクタのビット数の設定に用います．このビット数にはパリティ・ビット等の付加ビットは入りません．

プログラムしたキャラクタ長が8ビットより少ない場合は，上位桁のデータが無効になります．無効になったビットは，読み出し時は「0」になり，書き込み時は無視されます．

▶ D_4〜D_5：P_0〜P_1

パリティ・ビットの発生(送信)やチェック(受信)機能を制御します．パリティ発生/チェックは，キャラクタ・ビットとパリティ・ビットを合わせたビットの中で「1」であるビットの数が偶数(偶数パリティ)，または奇数(奇数パリティ)になるように，パリティ・ビットを発生/チェックします．

◈ 調歩同期モード特有のビット

▶ D_6〜D_7：ST_0〜ST_1

送信時に付加するストップ・ビットの長さの指定に用います．受信動作には影響を与えません(データ受信時には1ビットのストップ・ビットのみチェックされる)．

〈図 2-123 (a)〉 調歩同期モードにするためのモード・ワード

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
ST_1	ST_0	P_1	P_0	L_1	L_0	B_1	B_0

B_1	B_0	ボーレート
0	1	1倍クロック
1	0	16倍クロック
1	1	64倍クロック

L_1	L_0	キャラクタ長
0	0	5ビット
0	1	6ビット
1	0	7ビット
1	1	8ビット

P_1	P_0	パリティ発生/チェック
×	0	パリティなし
0	1	奇数パリティ
1	1	偶数パリティ

ST_1	ST_0	送信ストップ・ビット数
0	1	1ビット
1	0	$1\frac{1}{2}$ビット
1	1	2ビット

× : don't care

◈ 同期モード特有のビット

▶ D_6 : EXSYNC

同期検出方法を選択します．外部同期検出にプログラムした場合には，キャラクタ同期のための同期キャラクタの受信は必要ありません．

▶ D_7 : SSC

同期キャラクタ数を決めます．モード・ワードの次にSSCビットによって設定された数の同期キャラクタを書き込みます．

◈ コマンド設定(32H/ライト)

コマンド設定のフォーマットを図2-124に示します．

▶ D_0 : TxEN

送信の許可/禁止を指示します．送信禁止(TxEN＝0)にすると，その時点で書き込まれているデータをすべて送出してから送信を停止します．

▶ D_1 : DTR

8251の汎用出力ポートの制御用です．「DTR＝1」ならばRS-232-CコネクタのDTR端子は－Vになり，「DTR＝0」ならば＋Vになります．

▶ D_2 : RxEN

受信の許可/禁止を指示します．RxEN＝0で受信禁止です．同期モードの場合，受信禁止を行うと比同期状態になります．

▶ D_3 : SBRK

ブレーク信号の送出用で，SBRK＝1のとき現在送出中のデータを無効にして，8251のTxD出力をLレベルにします．SBRK＝0でブレーク状態が解除さ

〈図 2-123 (b)〉 同期モードにするためのモード・ワード

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SSC	EXSYNC	P_1	P_0	L_1	L_0	0	0

L_1	L_0	キャラクタ長
0	0	5ビット
0	1	6ビット
1	0	7ビット
1	1	8ビット

P_1	P_0	パリティ発生/チェック
×	0	パリティなし
0	1	奇数パリティ
1	1	偶数パリティ

EXSYNC	同期検出方法
0	内部同期検出
1	外部同期検出

SSC	同期キャラクタ数
0	2(BSC)
1	1

× : don't care

〈図 2-124〉 コマンド・ワード

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
EH	SRES	RTS	ECL	SBRK	RxEN	DTR	TxEN

TxEN	送信イネーブル
0	送信禁止
1	送信許可

DTR	DTR端子制御
0	DTR＝1
1	DTR＝0

RxEN	受信イネーブル
0	受信禁止
1	受信許可

SBRK	センド・ブレーク
0	TxDATA端子通常動作
1	TxDATA＝0

ECL	エラー・クリア
0	No Operation
1	エラー・フラグ・クリア

RTS	RTS端子制御
0	RTS＝1
1	RTS＝0

SRES	ソフトウェア・リセット
0	No Operation
1	リセット動作

EH*	エンタ・ハント・フェーズ
0	No Operation
1	ハント・フェーズに入る

* : EH(D_7) ビットは同期モードでのみ有効で，調歩同期モードではDon't careとなる．

れます．なお，この機能は送信禁止状態でも有効です．

▶ D_4 : ECL

8251のエラー・ステータス(PE，OE，FE)のクリアを行います．ER＝1でエラーで，クリア「0」されます．ハント・フェーズに入るとき(EH＝1)や受信許可(RxEN＝1)にするときは，同時に「ECL＝1」にします．

▶ D_5 : RTS

8251の汎用出力ポートの制御用です．「RTS＝1」ならばRS-232-CコネクタのDTR端子は－Vになり，

「RTS=0」ならば+Vになります．一般的には，受信データの抑制(フロー操作)に使われます．

▶ D_6：SRES

8251をソフトウェア・リセットさせます．IR=1でリセットし，モード設定待ち状態になります．

▶ D_7：EH

ハント・フェーズとは，同期モードを確立するために，RxDのレベルの変化を待っている状態です．ハント・フェーズに入るときはこのビットを「1」にします．このとき受信許可のRxENビットも「1」にしてください．同期キャラクタを検出し，同期を取ると，自動的にハント・フェーズから抜けてデータの受信が始ります．調歩同期モードでは使用しません．

◈ コントロール・ワード書き込みに関する注意

CPUがLSIに対して書き込みを行ってから，次の書き込みを行うまで，十分なタイミングを取る必要があります(書き込み回復時間)．

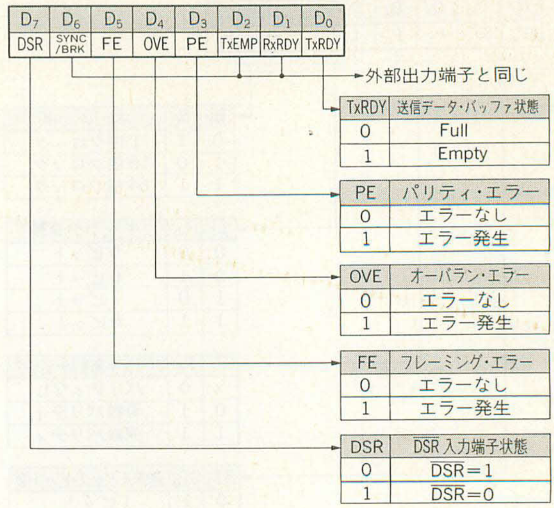
リセット・コマンド実行と，次のモード指定までには，最低$6t_{cyk}$分の時間をあけなくてはなりません．それ以外のコマンドとコマンドの間は，調歩同期モードで$8t_{cyk}$，同期モードで$16t_{cyk}$必要です($1t_{cyk}$は8251のCLK，1サイクル時間で，2.4576 MHzの場合では$1t_{cyk}$=約0.4μsになる)．

初期化の場合の書き込み回復時間は図2-122に示したとおりになります．

◈ ステータス読み出し(32H/リード)

ステータスは図2-125に示したとおりです．

〈図2-125〉ステータス



▶ D_0：TxRDY

送信データ・バッファ状態を示します．TxRDY=0でバッファにデータがあることを示し，この状態ではデータを送出できません．

TxRDYの割り込み要求端子の状態と同じです．この割り込みはマスク・セット(35H)で割り込み許可/禁止を設定できます．

▶ D_1：RxRDY

RxRDY=1でデータを受信したことを示します．

RxRDYの割り込み要求端子の状態と同じです．この割り込みはマスク・セット(35H)で割り込み許可/禁止を設定できます．

▶ D_2：TxEMP

送信データ・バッファ(第2バッファ)とトランスミッタ内の送信バッファ(第1バッファ)が共に空であることを示します．

TxEMPの割り込み要求端子の状態と同じです．この割り込みはマスク・セット(35H)で割り込み許可/禁止を設定できます．

▶ D_3：PE

パリティ・エラーの発生を示します．エラーがあれば「1」，なければ「0」になります．エラーが発生しても8251の動作は停止しません．ER=1でエラーはクリアされます．

▶ D_4：OVE

オーバラン・エラーの発生を示します．CPUが受信データの読み出しに遅れたときに「1」になります．エラーが発生しても8251の動作は停止しません．ER=1でエラーはクリアされます．

▶ D_5：FE

フレーミング・エラーの発生を示します．ストップ・ビットが検出されなかったときに「1」になります．エラーが発生しても8251の動作は停止しません．ER=1でエラーはクリアされます．

▶ D_6：SYNC/BRK

調歩同期モードでは，ブレーク信号(RxDが2キャラクタ以上の時間「0」になった場合)を受信したときに「1」になります．同期モード・内部同期検出の場合は，同期キャラクタを検出したときに「1」になります．

▶ D_6：DSR

汎用入力ポートのDSRの状態を示します．RS-232-CコネクタのDTR端子が−Vのときに「DSR=1」になり，+Vならば「DSR=0」になります．

◈ リード・シグナル(33H/リード)

CD，CS(CTS)，CIの読み出しには，システム・ポートのポートBを使用しています．

- D_5：CD　　RS-232-C
- D_6：CS(CTS)　　RS-232-C
- D_7：CI　　RS-232-C

RS-232-C用に使用されている8251では読み取ることのできない，CI，(CTS)CS，CDの信号を読み

〈図 2-126〉拡張 RS-232-C の I/O アドレス

LSI	命　令	I/O ポート・アドレス		R/W	データ								備　考
		CH_2	CH_3		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
μPD8251	モード(A)	B3	BB	W	S_2	S_1	EP	PEN	L_2	L_1	×	×	μPD8251 動作モードの設定(非同期)
	モード(B)	B3	BB	W	SCS	ESD	EP	PEN	L_2	L_1	0	0	μPD8251 動作モードの設定(同期)
	コマンド	B3	BB	W	EH	IR	RS	RST	SBR	REN	ER	TEN	
	ステータス	B3	BB	R	DR	SYN	FE	OE	PE	TE	RRDY	TRDY	
	データ・リード	B1	B9	R	RD_8	RD_7	RD_6	RD_5	RD_4	RD_3	RD_2	RD_1	
	データ・ライト	B1	B9	W	SD_8	SD_7	SD_6	SD_5	SD_4	SD_3	SD_2	SD_1	
	マスク・セット	B0	B2	W	×	×	×	×	×	TXR	TXE	RXR	
	リード・シグナル	B0	B2	R	$\overline{CI}$	$\overline{CS}$	$\overline{CD}$	×	×	×	×	×	
	割り込みレベル・センス	B0	B2	R	×	×	×	×	×	×	IR_1	IR_2	

取ることができます．RS-232-C の各信号が，+V (OFF)のときや開放されているときに，対応するビットが“1”になります．−V(ON)のときは“0”です．

◈ マスク・セット(35H/ライト)

RS-232-C の割り込みマスク許可/禁止には，システム・ポートのポート C を使用しています．

- D_0：RxRDY 割り込みイネーブル
- D_1：TxEMPT 割り込みイネーブル
- D_2：TxRDY 割り込みイネーブル

RS-232-C 用割り込みのマスク用です．“1” で割り込み禁止，“0” で割り込み可能になります．

拡張 RS-232-C インターフェース

▶使用 LSI
μPD8251A 相当×2
▶ I/O アドレス
B1H，B3H(チャネル#2)
B9H，BBH(チャネル#3)
B0H，B2H(割り込みセンス，リード・シグナル)
▶使用割り込み
INT_{0-6}(任意に設定できる)

オプションで拡張スロットに RS-232-C インターフェースを増設できます．NEC 純正として PC9861/K というオプションがありますが，いくつかのサード・パーティからも似たようなボードが販売されています．

使用 LSI に，μPD8251A 相当を二つ使い，内蔵 RS-232-C インターフェースと，ほぼ同様の手順で操作できます．違いは，ボーレート・ジェネレータを拡張ボード上に持ちハードウェア(ディップ・スイッチ)で変更することと，割り込みレベルをハードウェア(ディップ・スイッチ)で変更できることです．

〈図 2-127〉
割り込みレベル

IR_1	IR_2	INT レベル	
		CH_2	CH_3
0	0	INT_0	INT_0
0	1	INT_1	INT_4
1	0	INT_2	INT_5
1	1	INT_3	INT_6

AIWA の B98 - 01 では，××D1H，××D3H，××D5H，××D7H(××は任意に設定できる)で，ボーレートの変更や，自己診断機能の ON-OFF が設定できます．拡張 RS-232-C の I/O アドレス一覧を図 2-126 に示します．

◈ リード・シグナル/割り込みレベル・センス (CH_2：B0H/CH_3：B2H)

▶ D_0〜D_1：IR_1〜IR_2

ディップ・スイッチで設定した割り込みレベルを読み出せます．B98-01 では「$D_2(IR_3)$」も使用されています．割り込みレベルを図 2-127 に示します．

- D_5：CD　　RS-232-C
- D_6：CS(CTS)　　RS-232-C
- D_7：CI　　RS-232-C

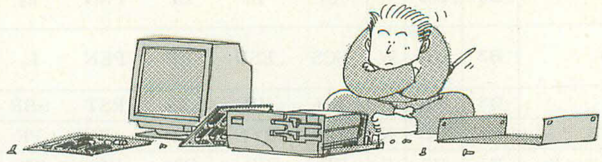
RS-232-C 用に使用されている 8251 では読み取ることのできない，CI，(CTS)CS，CD の信号を読み取ることができます．RS-232-C の各信号が，+V (OFF)のときや開放されているときに，対応するビットが“1”になります．−V(ON)のときは“0”です．

◈ マスク・セット(CH_2：B0H/CH_3：B2H)

- D_0：RxRDY 割り込みイネーブル
- D_1：TxEMPT 割り込みイネーブル
- D_2：TxRDY 割り込みイネーブル

RS-232-C 用割り込みのマスク用です．“1” で割り込み禁止，“0” で割り込み可能になります．

3 拡張スロットの信号と使い方



PC 98 シリーズの拡張スロットの詳細

◈ 拡張スロットの種類

PC98シリーズの拡張バスは，大きく分けると，デスクトップ系，ノート系，NESAバス等があります．そのうち最も一般的なものは，初代PC9801からの流れをくむデスクトップ系の拡張スロットで，拡張基板の外形や，寸法は初代から現在まで変わりません．

拡張スロットの信号線(98バス)は全部で50本あり，データ・バス16ビット，アドレス・バス24ビット(8086/V30系は20ビット)等が出ています．

NECのノート系は，拡張基板が挿入できるスロットではなく，拡張バス・コネクタが装備されています．このコネクタは，従来のデスクトップ系とほぼ同じ信号線に加えて，増設用FDDやCRT関連の信号が追加され，合計110本あります．専用のI/O拡張ユニットを使用することで，デスクトップ用の拡張基板を使用することもできます．

EPSONのラップトップ，BOOK系の機種には，Lスロットと呼ばれる拡張スロットが搭載されています．デスクトップ系のスロットとは形状や寸法が小さくなっていますが，信号線は50本で，98バスとほぼ同等な規格になっています．

EPSONのNOTE系の機種のバスは，EPSON独自規格の80本バスと，NECノートと同一の110本バスの2種類があります．PC286NOTEexecutiveから，PC386NOTE WRまでは80本バスで，PC386NOTE AE以降には110本バスが搭載されています．110本バスはNECのノートのバスとの互換性があります．

80本のバスと110本バスの違いは，増設FDDの信号がない，アドレス・バスのA_{19}～A_{23}がない，CRT出力信号の違い等があります．80本バスを持つ機種には増設FDDコネクタが別に付いています．

NESAバスは，PC-H98シリーズに搭載されているバスで，バス幅を従来の16ビットから32ビットに拡張し，新しいアーキティクチャによる高性能なバスです．PC-H98シリーズにしか搭載されず，使用ユーザが増えないためか，拡張基板の価格が割高なようです．

◈ 拡張スロットの数

拡張スロットの数は機種によって違います．初代PC9801やPC9801Eでは，FDD/HDDインターフェースが標準搭載されていなかったためもあって5～6個付いていましたが，一般的なデスクトップは4個が標準になっています．デスクトップでもローコスト版の機種では3個に減らされているものもありますし，小型化された機種では2個，PC286C(EPSON)では1個しかありません．

◈ 拡張基板の形状と寸法

拡張基板の外形，寸法は図3-1のようになっています．バス・コネクタは，カード・エッジ・コネクタで，基板にはカード・エッジ・パターンが2.54 mmピッチで50個×両面で合計100個あります．

拡張基板の端子は，はんだ面(裏面)がA，部品面(表面)がBで，A_{01}～A_{50}とB_{01}～B_{50}となっています．

スロット一つの高さは25 mmありますが，基板上面からのスペースは20 mmほどしかありません．また，基板の抜き差し用にカード・プラが取り付けられます．

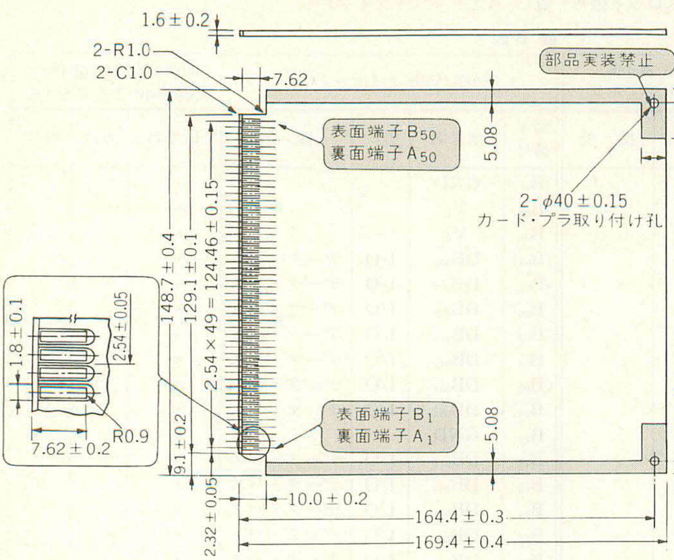
参考にEPSON・Lスロット基板の外形と寸法を図3-2に示します．

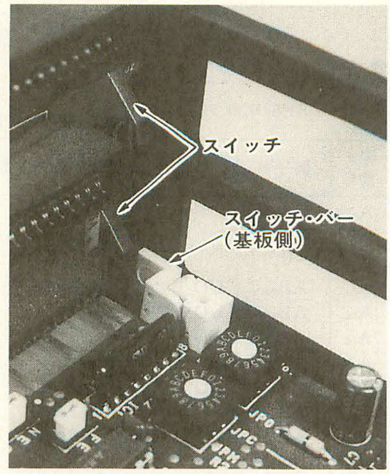
◈ 機種による拡張バスの違い

PC98シリーズのアドレス・バスは2種類あります．8086/V30等の1Mバイトしかアドレス空間(20ビット)を持たないCPU用バスと，80286以降の16Mバイト以上のアドレス空間(24ビット)を持つCPU用バスです．

拡張スロットには，8086/V30用の機種にも24ビット分のアドレス端子が用意されてますが，使用されてはいません．そのために，これらの機種用に設計された拡張基板では，上位4ビット分のアドレス(A_{20}

〈図3-1〉拡張基板の外形と寸法





〈写真3-1〉拡張基板側のスイッチ・バー

～A_{23})をデコードしていない可能性があります.

そこで，1Mバイト以下のメモリ空間(A_{20}～A_{23}がすべて“0”)になったときだけA19を有効にすることで，上位4ビット分のアドレスをデコードしていないボードでも，1Mバイト以上のメモリ空間にイメージが出ないようにするためのスイッチがあります.

このスイッチは拡張スロットのバス・コネクタの横に取り付けられていて，拡張基板側のスイッチ・バーによって，基板挿入時に自動的に押されるようになっています(**写真3-1**).

拡張スロットの信号線

図3-3にバス・スロット信号一覧を示します.

▶ AB_{001}～AB_{191}&AB_{201}～AB_{231}：アドレス・バス

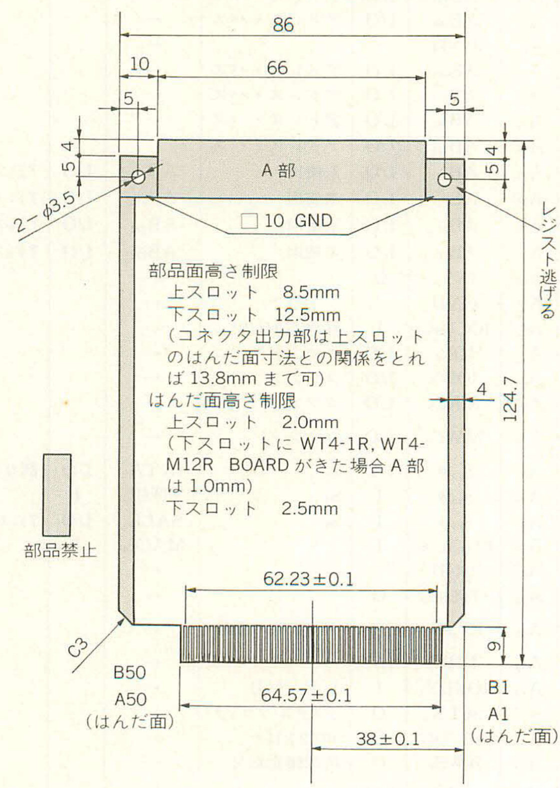
アドレス・バスは24ビットで構成され，16Mバイトまでのメモリ空間を持っています．ただし，8086/V30等のCPU自身が1Mバイトのメモリ空間しか持っていない機種では，AB_{201}～AB_{231}は未使用です.

本体内部のCPUやDMACが動作中は出力ですが，CPUがディス・イネーブルされたり，外部DMAC等が動作する場合は入力になります.

▶ BHE_0：バス・ハイ・エネーブル

データ・バスの上位バイトのアクセスを示します．データ・バスは16ビット幅ですが，アドレス・バスは8ビットごとにアクセスできるようになっています．16ビット分同時にアクセスしようとすると，アドレス・バスの最下位信号であるAB_{001}(A_0)信号を使っていてはできません．そこで，データ・バスの上位8ビットをアクセスするのに便利なようにBHE_0があります．AB_{001}と逆論理の信号のような動作になります.

〈図3-2〉EPSON・Lスロット基板の外形と寸法



▶ DB_{001}～DB_{151}：データ・バス

データ・バスは16ビットで構成されています．下位8ビットは偶数アドレス，上位8ビットは奇数アドレスに相当しますので，8ビット幅のI/Oポートを設計する場合は注意が必要です.

▶ IOR_0：I/Oリード

〈図 3-3 (a)〉 バス・スロット信号一覧(デスクトップ・タイプ)

● A 面

8086/V30 タイプ・バス				80286/80386/80486/Pentium タイプ・バス		
端子番号	信号名	方向	機　能	信号名	方向	機　能
A_{01}	GND			←		
A_{02}	V_1			←		
A_{03}	V_2			←		
A_{04}	AB_{001}	I/O	アドレス・バス	←		
A_{05}	AB_{011}	I/O	アドレス・バス	←		
A_{06}	AB_{021}	I/O	アドレス・バス	←		
A_{07}	AB_{031}	I/O	アドレス・バス	←		
A_{08}	AB_{041}	I/O	アドレス・バス	←		
A_{09}	AB_{051}	I/O	アドレス・バス	←		
A_{10}	AB_{061}	I/O	アドレス・バス	←		
A_{11}	GND			←		
A_{12}	AB_{071}	I/O	アドレス・バス	←		
A_{13}	AB_{081}	I/O	アドレス・バス	←		
A_{14}	AB_{091}	I/O	アドレス・バス	←		
A_{15}	AB_{101}	I/O	アドレス・バス	←		
A_{16}	AB_{111}	I/O	アドレス・バス	←		
A_{17}	AB_{121}	I/O	アドレス・バス	←		
A_{18}	AB_{131}	I/O	アドレス・バス	←		
A_{19}	AB_{141}	I/O	アドレス・バス	←		
A_{20}	AB_{151}	I/O	アドレス・バス	←		
A_{21}	GND			←		
A_{22}	AB_{161}	I/O	アドレス・バス	←		
A_{23}	AB_{171}	I/O	アドレス・バス	←		
A_{24}	AB_{181}	I/O	アドレス・バス	←		
A_{25}	AB_{191}	I/O	アドレス・バス	←		
A_{26}	AB_{201}	I/O	未使用	AB_{201}	I/O	アドレス・バス
A_{27}	AB_{211}	I/O	未使用	AB_{211}	I/O	アドレス・バス
A_{28}	AB_{221}	I/O	未使用	AB_{221}	I/O	アドレス・バス
A_{29}	AB_{231}	I/O	未使用	AB_{231}	I/O	アドレス・バス
A_{30}	INT_0	O		←		
A_{31}	GND			←		
A_{32}	$IOCHK_0$	I	外部 NMI(2)	←		
A_{33}	IOR_0	I/O	コマンド	←		
A_{34}	IOW_0	I/O	コマンド	←		
A_{35}	MRC_0	I/O	コマンド	←		
A_{36}	MWC_0	I/O	コマンド	←		
A_{37}	S_{00} *	I/O	S_0	$INTA_0$	I/O	割り込み
A_{38}	S_{10} *	I	S_1	$NOWAIT_0$	I	
A_{39}	S_{20} *	I	S_2	$SALE_1$	I/O	アドレス・ラッチ
A_{40}	LOCK *	I		$MACS_0$	I	
A_{41}	GND			←		
A_{42}	$CPUENB_{10}$	O		←		
A_{43}	$RFSH_0$	O	リフレッシュ信号	←		
A_{44}	BHE_0	I/O		←		
A_{45}	$IORDY_1$	I	レディ信号	←		
A_{46}	$SCLK_1$	O	システム・クロック	←		
A_{47}	$S18CLK_1$	O	307.2 kHz	←		
A_{48}	$POWER_0$	O	電源確定信号	←		
A_{49}	+5 V			←		
A_{50}	+5 V			←		

● B 面

8086/V30 タイプ・バス				80286/80386/80486/Pentium タイプ・バス		
端子番号	信号名	方向	機　能	信号名	方向	機能
B_{01}	GND			←		
B_{02}	V_1			←		
B_{03}	V_2			←		
B_{04}	DB_{001}	I/O	データ・バス	←		
B_{05}	DB_{011}	I/O	データ・バス	←		
B_{06}	DB_{021}	I/O	データ・バス	←		
B_{07}	DB_{031}	I/O	データ・バス	←		
B_{08}	DB_{041}	I/O	データ・バス	←		
B_{09}	DB_{051}	I/O	データ・バス	←		
B_{10}	DB_{061}	I/O	データ・バス	←		
B_{11}	GND			←		
B_{12}	DB_{071}	I/O	データ・バス	←		
B_{13}	DB_{081}	I/O	データ・バス	←		
B_{14}	DB_{091}	I/O	データ・バス	←		
B_{15}	DB_{101}	I/O	データ・バス	←		
B_{16}	DB_{111}	I/O	データ・バス	←		
B_{17}	DB_{121}	I/O	データ・バス	←		
B_{18}	DB_{131}	I/O	データ・バス	←		
B_{19}	DB_{141}	I/O	データ・バス	←		
B_{20}	DB_{151}	I/O	データ・バス	←		
B_{21}	GND			←		
B_{22}	+12 V			←		
B_{23}	+12 V			←		
B_{24}	IR_{31}	I	INT_0	←		
B_{25}	IR_{51}	I	INT_1	←		
B_{26}	IR_{61}	I	INT_2	←		
B_{27}	IR_{91}	I	INT_3	←		
B_{28}	IR_{101}/IR_{111}	I	INT_{41}/INT_{42}(1)	IR_{101}	I	INT_{41}
B_{29}	IR_{121}	I	INT_5	←		
B_{30}	IR_{131}	I	INT_6	←		
B_{31}	GND			←		
B_{32}	−12 V			←		
B_{33}	−12 V			←		
B_{34}	$RESET_0$	O	/RESET	←		
B_{35}	$DACK_{00}$	O	HDC	←		
B_{36}	$DACK_{30}/DACK_{20}$	O	AUX(1)	$DACK_{30}$	O	
B_{37}	DRQ_{00}	I	HDC	←		
B_{38}	DRQ_{30}/DRQ_{20}	I	AUX(1)	DRQ_{30}	I	
B_{39}	$WORD_0$	I		←		
B_{40}	$CPKILL_0$ *	I		$EXHRQ_{10}$	I	
B_{41}	GND			←		
B_{42}	$RQGT_0$ *	I/O	バスの解放要求	$EXHLA_{10}$	O	
B_{43}	$DMATC_0$	O	END OF PROCESS	←		
B_{44}	NMI_0	O		←		
B_{45}	MWE_0	I/O		←		
B_{46}	$HLDA_{00}$ *	O		$EXHLA_{20}$	O	
B_{47}	HRQ_{00} *	I		$EXHRQ_{20}$	I	
B_{48}	$DMAHLD_0$ *	I		$SUBSRQ_1$	O	
B_{49}	+5 V			←		
B_{50}	+5 V			←		

* 8086/V30 タイプ・バス
PC9801/E/F/M/U/VF/VM/UV/CV/UF/UR/PC98DO/DO$^+$
PC9801VM21(スロット#1)/VX2(スロット#1)　　PC9801LV21(PC9801LV-8)

* 80286/80386/80486/Pentium タイプバス
PC9801VX21/UX/RA/RX/EX/ES/RS/T/DX/DS/DA/FA/FS/FX/US/BA/BX
PC9801VM21(スロット#2.3.4)/VX2(スロット#2.3.4)　　PC9801LS/LX(PC9801LV-8)
PC9821/Ap/As/Ae/Ce/Af　　PC98XA/XL/XL2/RL/GS
EPSON-PC DESKTOP　　EPSON-PC L-SLOT(A27-A29 は未接続, PC286L/LE/LF/LP は若干異なる)

〈図 3-3 (b)-1〉 外部拡張コネクタ 110 ピン・タイプ(ノート・タイプ)

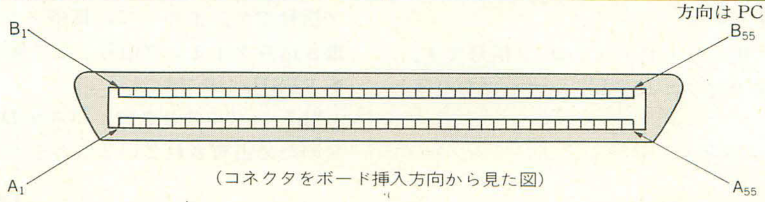
端子番号	信号名	方向	意　味
A_1	+5 V	—	+5 V 電源
A_2	+5 V	—	+5 V 電源
A_3	SD_{14}	I/O	データ・バス
A_4	SA_{12}	I/O	データ・バス
A_5	SA_{10}	I/O	データ・バス
A_6	SA_8	I/O	データ・バス
A_7	SA_6	I/O	データ・バス
A_8	SA_4	I/O	データ・バス
A_9	GND	—	グラウンド
A_{10}	SD_2	I/O	データ・バス
A_{11}	SD_0	I/O	データ・バス
A_{12}	SA_{19}	O	アドレス・バス
A_{13}	SA_{17}	O	アドレス・バス
A_{14}	SA_{15}	O	アドレス・バス
A_{15}	SA_{13}	O	アドレス・バス
A_{16}	SA_{11}	O	アドレス・バス
A_{17}	GND	—	グラウンド
A_{18}	SA_9	O	アドレス・バス
A_{19}	SA_7	O	アドレス・バス
A_{20}	SA_5	O	アドレス・バス
A_{21}	SA_3	O	アドレス・バス
A_{22}	SA_1	O	アドレス・バス
A_{23}	GND	—	グラウンド
A_{24}	SMRD	O	メモリ・リード・コマンド
A_{25}	GND	—	グラウンド
A_{26}	SIOR	O	I/O リード・コマンド
A_{27}	NC	—	未接続
A_{28}	INT_5	I	INT_5(拡張用)
A_{29}	INT_2	—	INT_2(未接続)
A_{30}	NC	—	未接続
A_{31}	$DACK_{30}$	O	DMA アクノリッジ・チャネル 3
A_{32}	INT_4	—	INT_4(未接続)
A_{33}	DRQ_{30}	I	DMA リクエスト・チャネル 3
A_{34}	WGATE	O	ライト・ゲート信号
A_{35}	NC	—	未接続
A_{36}	SALE	O	アドレス・ラッチ信号
A_{37}	SA_{22}	O	アドレス・バス
A_{38}	SCLK	O	システム・クロック
A_{39}	GND	—	グラウンド
A_{40}	IORDY	I	レディ信号
A_{41}	NC	—	未接続
A_{42}	$CPUE_0$	O	CPU 動作中
A_{43}	$POWER_0$	O	電源確定信号
A_{44}	INT_6	I	INT_6
A_{45}	SA_{21}	I	アドレス・バス
A_{46}	NC	—	未接続
A_{47}	SA_{23}	O	アドレス・バス
A_{48}	STEP /VSYNC	O	ステップ信号/垂直同期信号
A_{49}	SSEL /HSYNC	O	サイド・セレクト/水平同期信号
A_{50}	DS_2	—	ドライブ・セレクト 2
A_{51}	NC	—	未接続
A_{52}	RDATA	I	リード・データ
A_{53}	RGBSEL	I	FDD/CRT 選択
A_{54}	TRK_0	I	トラック 0
A_{55}	WPRT	I	ライト・プロテクト

端子番号	信号名	方向	意　味
B_1	+5 V	—	+5 V 電源
B_2	SD_{15}	I/O	データ・バス
B_3	SD_{13}	I/O	データ・バス
B_4	SD_{11}	I/O	データ・バス
B_5	SD_9	I/O	データ・バス
B_6	SD_7	I/O	データ・バス
B_7	SD_5	I/O	データ・バス
B_8	SD_3	I/O	データ・バス
B_9	MFM	O	MFM 信号
B_{10}	SD_1	I/O	データ・バス
B_{11}	SBHE	—	バス・ハイ・イネーブル
B_{12}	SA_{18}	O	アドレス・バス
B_{13}	SA_{16}	O	アドレス・バス
B_{14}	SA_{14}	O	アドレス・バス
B_{15}	SA_{12}	O	アドレス・バス
B_{16}	SA_{10}	O	アドレス・バス
B_{17}	SYNC	O	SYNC 信号
B_{18}	SA_8	O	アドレス・バス
B_{19}	SA_6	O	アドレス・バス
B_{20}	SA_4	O	アドレス・バス
B_{21}	SA_2	O	アドレス・バス
B_{22}	SA_0	O	アドレス・バス
B_{23}	GND	—	グラウンド
B_{24}	SMWR	O	メモリ・ライト・コマンド
B_{25}	GND	—	グラウンド
B_{26}	SIOW	O	I/O ライト・コマンド
B_{27}	$IOCHK_0$	I	外部 NMI 要求信号
B_{28}	INT_3	I	INT_3(HDD)
B_{29}	$DACK_{00}$	O	DMAアクノリッジ・チャネル0
B_{30}	NC	—	未接続
B_{31}	DRQ_{00}	I	DMA リクエスト・チャネル 0
B_{32}	NC	—	未接続
B_{33}	$INTA_0$	O	割り込みアクノリッジ信号
B_{34}	NC	—	未接続
B_{35}	NC	—	未接続
B_{36}	SA_{20}	O	アドレス・バス
B_{37}	$REST_0$	O	システム・リセット
B_{38}	S18CLK	O	約 307.2 kHz
B_{39}	GND	—	グラウンド
B_{40}	INT_0	I	INT_0(拡張用)
B_{41}	INT_1	I	INT_1(拡張用)
B_{42}	$WORD_0$	I	DMA ワード転送要求信号
B_{43}	$RFSH_0$	O	リフレッシュ信号
B_{44}	$DMATC_0$	O	DMA End of Process
B_{45}	HID /AG	O O	ヘッドロード信号/アナログ緑色ビデオ信号
B_{46}	NMI_0	O	NMI 出力信号
B_{47}	DIR /AR	O O	ディレクション信号/アナログ青色ビデオ信号
B_{48}	WINDOW	I	ウィンドウ信号
B_{49}	WDATA /AB	O O	ライト・データ信号/アナログ赤色ビデオ信号
B_{50}	GND	—	グラウンド
B_{51}	NC	—	未接続
B_{52}	READY	I	ドライブ・レディ
B_{53}	DS_3	O	ドライブ・セレクト 3
B_{54}	INDEX	I	インデックス信号
B_{55}	$SMWE_0$	O	メモリ・ライト・イネーブル

方向は PC 本体を基準としたもの

〈図 3-3 (b)-2〉

110 ピン・コネクタ



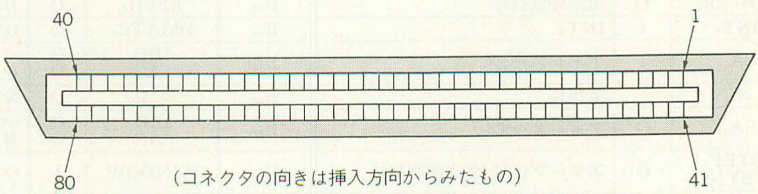
(コネクタをボード挿入方向から見た図)

〈図 3-3 (b)-3〉 外部拡張コネクタ 80 ピン・タイプ (エプソンノート・タイプ)

端子番号	信号名	方向	意味
1	SA_{12}	I/O	アドレス・バス
2	SA_{11}	I/O	アドレス・バス
3	SA_{10}	I/O	アドレス・バス
4	SA_9	I/O	アドレス・バス
5	SA_8	I/O	アドレス・バス
6	SA_7	I/O	アドレス・バス
7	SA_6	I/O	アドレス・バス
8	SA_5	I/O	アドレス・バス
9	SA_4	I/O	アドレス・バス
10	SA_3	I/O	アドレス・バス
11	SA_2	I/O	アドレス・バス
12	SA_1	I/O	アドレス・バス
13	SA_0	I/O	アドレス・バス
14	SD_{15}	I/O	データ・バス
15	SD_{14}	I/O	データ・バス
16	SD_{13}	I/O	データ・バス
17	SD_{12}	I/O	データ・バス
18	SD_{11}	I/O	データ・バス
19	SD_{10}	I/O	データ・バス
20	SD_9	I/O	データ・バス
21	SD_8	I/O	データ・バス
22	SD_7	I/O	データ・バス
23	SD_6	I/O	データ・バス
24	SD_5	I/O	データ・バス
25	SD_4	I/O	データ・バス
26	SD_3	I/O	データ・バス
27	SD_2	I/O	データ・バス
28	SD_1	I/O	データ・バス
29	SD_0	I/O	データ・バス
30	DTCK	O	ドット・クロック (21.052 MHz)
31	HSYNC	O	水平同期信号
32	VSYNC	O	垂直同期信号
33	GRN_1	O	表示データ緑 1
34	RED_1	O	表示データ赤 1
35	GRN_2	O	表示データ緑 2
36	RED_2	O	表示データ赤 2
37	GRN_3	O	表示データ緑 3
38	RED_3	O	表示データ赤 3
39	GRN_4	O	表示データ緑 4
40	RED_4	O	表示データ赤 4
41	SA_{19}	I/O	アドレス・バス
42	GND	—	グラウンド
43	SA_{18}	I/O	アドレス・バス
44	+5 V	—	+5 V 電源
45	SA_{17}	I/O	アドレス・バス
46	INT_0	I	INT_0
47	SA_{16}	I/O	アドレス・バス
48	INT_1	I	INT_1
49	SA_{15}	I/O	アドレス・バス
50	GND	—	グラウンド
51	SA_{14}	I/O	アドレス・バス
52	+5 V	—	+5 V 電源
53	SA_{13}	I/O	アドレス・バス
54	INT_2	I	INT_2*
55	SIOR	I/O	I/O リード・コマンド
56	INT_3	I	INT_3
57	SIOW	I/O	I/O ライト・コマンド
58	GND	—	グラウンド
59	SMRD	I/O	メモリ・リード・コマンド
60	+5 V	—	+5 V 電源
61	SMWR	I/O	メモリ・ライト・コマンド
62	DAK_{00}	O	DMA アクノリッジ・チャネル 0
63	$CPUE_0$	O	CPU イネーブル
64	DRQ_{00}	I	DMA リクエスト・チャネル 0
65	IORDY	I	ウェイト要求信号
66	GND	—	グラウンド
67	SCLK	O	システム・クロック
68	+5 V	—	+5 V 電源
69	SBHE	I/O	バス・ハイ・イネーブル
70	$DMATC_0$	O	DMA ターミナル・カウント
71	SYNC	O	VSYNC と HSYNC の 排他的論理和
72	(NC)	—	(未使用)
73	BLE_1	O	表示データ青 1
74	GND	—	グラウンド
75	BLE_2	O	表示データ青 2
76	+5 V	—	+5 V 電源
77	BLE_3	O	表示データ青 3
78	$REST_0$	O	システム・リセット
79	BLE_4	O	表示データ青 4
80	GND	—	グラウンド

* PC-386NOTE W/WR は NC，方向は PC 本体を基準としたもの

〈図 3-3 (b)-4〉
80 ピン外部拡張コネクタ



(コネクタの向きは挿入方向からみたもの)

I/O アクセス用のリード・ストローブ信号です．I/O リードするサイクルで “L” レベルになります．

▶ IOW_0 : I/O ライト

I/O アクセス用のライト・ストローブ信号です．I/O ライトするサイクルで “L” レベルになります．

▶ MRC_0 : メモリ・リード

メモリ・アクセス用のリード・ストローブ信号です．メモリ・リードするサイクルに “L” レベルになります．

▶ MWC_0 : メモリ・ライト

メモリ・アクセス用のライト・ストローブ信号です．メモリ・ライトするサイクルで “L” レベルになります．

▶ MWE_0 : メモリ・ライト・イネーブル

MWC_0よりも遅れたタイミングのライト・ストローブ信号です．主として，拡張メモリに対する DRAM 書き込みタイミング信号として使われます．

▶ $RFSH_0$: リフレッシュ

“L” レベルのときに，バスが DRAM のリフレッシュのため占有されていることを示します．このときは，

メモリに対して読み書きを行ってはいけません．

▶ IR_{31}～IR_{131}：割り込み要求信号(INT_0～INT_6)

外部からCPUに対してマスカブル割り込みをかけるときの信号です．8259(割り込みコントローラ)に接続されていて処理されます．PC98では，8259がエッジ・トリガ・モードで使用されているために，割り込み要求信号が，“L”から“H”へ変化したときに割り込みがかかります．

IR_{101}，IR_{111}はフロッピ・インターフェース用の割り込み要求信号で，同じ端子(B_{28})に割り当てられています．8086/V30の機種ではスロット番号が一番大きいスロットにIR_{111}が割り当てられ，その他のスロットにはIR_{101}が割り当てられています．80286以降の機種(PC98XAを除く)では，すべてIR_{101}が割り当てられ，IR_{111}は使用できません．

▶ $IOCHK_0$：NMI要求信号

CPUに対してノン・マスカブル割り込み(NMI)をかけるための入力端子です．NMIはソフトウェアから割り込みを禁止できないハードウェア割り込みで，拡張メモリのパリティ・エラー検出に使用されます．

▶ INT_0：(マスカブル)インタラプト

IR_{31}～IR_{131}までの割り込み要求信号に，割り込みコントローラ(8259)が応答したことを示します．

▶ NMI_0：(ノン・マスカブル)インタラプト

$IOCHK_0$(NMI要求信号)があったことを示します．

▶ $SCLK_1$：システム・クロック

拡張バスでのCPUのクロックです．8086ではデューティ比が2：1ですが，それ以外のCPUでは1：1になっています．

CPUクロック　8/16 MHz＝7.982 MHz
　　　　　　　10/12/20 MHz＝9.8304 MHz
　　　　　　　5 MHz＝4.9152 MHz

基本的には，上記のようにCPUクロックで，システム・クロックが決まりますが，中には，CPUクロックとは無関係に決定されている機種もあります．

▶ $S18CLK_1$：(307.2 kHz)

307.2 kHzのクロック信号です．シリアル回線用のボーレート・クロックとして使用すると便利です．

▶ $POWER_0$：電源確認信号

電源ON/OFF時にDC＋5 V電源電圧が，＋4.75 V以上になったときに有効になります．

▶ $RESET_0$：リセット信号

電源投入時にDC＋5 V電源電圧が，＋4.75 V以上になったときか，本体のリセット・スイッチが押されたときに，“L”レベルになります．

▶ DRQ_{00}～DRQ_{30}：DMA要求信号

DMAによってデータ転送を行うときに，I/OデバイスがDMAコントローラに対して行うDMA要求信号です．DMAコントローラは，この信号を見て，CPUにバス空け渡し(バス・ホールド)信号を出します．

DRQ_{20}，DRQ_{30}はフロッピ・インターフェース用のDMA要求信号で，同じ端子(B_{38})に割り当てられています．前述のIR_{101}，IR_{111}と同様に，バスの種類によって出ているスロット番号が違います．80286以降の機種ではDRQ_{30}が割り当てらています．

▶ $DACK_{00}$～$DACK_{30}$：DMAアクノレッジ信号

DMAが出したバス・ホールド信号にCPUが許可を与えたられたときに，DMA要求をしたデバイスに，DMAが使用可能になったことを知らせる信号です．

$DACK_{20}$，$DACK_{30}$はフロッピ・インターフェース用のDMAアクノレッジ信号で，同じ端子(B_{36})に割り当てられています．前述のIR_{101}，IR_{111}と同様にバスの種類によって，出ているスロット番号が違います．80286以降の機種では$DACK_{30}$が割り当てられています．

▶ $WORD_0$：ワード/バイト

内部DMAに接続するI/Oデバイスがワード転送かバイト転送かを示す信号です．ワード転送のときにDACK信号と同期させて，この信号を“L”にしなければなりません．ただし，ノーマル・モードでは未使用です．

▶ $DMTC_0$：DMAターミナル・カウント

DMA転送の転送時の終了バイトのときに“L”になります．

▶ $CPUENB_{10}$：CPUイネーブル信号

トランジスタ技術 SPECIAL No.35　好評発売中
B5判　160頁
定価1,600円
特集　C言語による回路シミュレータの製作
Quick Cでのプログラミングとフィルタ回路の解析
MICRO-CAPIIIやPSpiceなどの回路シミュレータではどういうアルゴリズムを使って電子回路を解析しているかを具体的に解説します．
CQ出版社 〒170 東京都豊島区巣鴨1-14-2 販売部☎(03)5395-2141 振替 00100-7-10665

〈図 3-4 (a)〉 バス・スロットのドライブ能力(8086/V30 CPU 機)

信号名	1スロット当たりの最大入力電流		外部ロジックがドライブするときの最小出力電流(注)	
	I_{IL} (mA)	I_{IH} (μA)	I_{OL} (mA)	I_{OH} (mA)
AB$_{101}$~AB$_{191}$	−0.8	40	12	−1.2
BHE$_0$				
DB$_{001}$~DB$_{151}$				
IOR$_0$				
IOW$_0$				
MRC$_0$				
MWC$_0$				
MWE$_0$			不可	
RFSH$_0$	全スロットで−0.8	全スロットで40	12	−1.2
IR$_{31}$~IR$_{131}$	—	—	8	−0.4
IOCHK$_0$	—	—		
INT$_0$	−0.8	40	不可	
NMI$_0$				
SCLK$_1$				
S18CLK$_1$				
POWER$_0$				
RESET$_0$				
DRQ$_{00}$, DRQ$_{30}$	—	—	8	−0.4
DACK$_{00}$, DACK$_{30}$	全スロットで−1.6*	全スロットで80*	不可	
WORD$_0$	—	—	不可	
DMATO$_0$	全スロットで−1.6*	全スロットで80*	8	−0.4
DMAHLD$_0$	—	—		
HRQ$_{00}$	—	—		
HRDA$_{00}$	−0.8	40	不可	
CPUENB$_{10}$				
IORDY$_0$	—	—	8	−0.4
S$_{00}$, S$_{10}$, S$_{20}$	全スロットで−0.4	全スロットで20		
RQ/GT$_0$	—	—		
LOCK$_0$	全スロットで−0.4	全スロットで20		
CPKILL$_0$	—	—		

注：外部ロジックはIR$_{31}$~IR$_{131}$，DRQ$_{00}$，DRQ$_{30}$を除きトライ・ステート出力であること．トライ・ステート・ハイ・インピーダンス時のリーク電流は20 μA以下とする

＊ PC9801では，$I_{IL}=-0.8$ mA，$I_{IH}=40\ \mu$A

CPUがバスを使用しているときに“L”になります．一般的には，アドレス・デコーダのイネーブル端子に接続して，CPUアクセス時のみに動作するように使います．

▶ IORDY$_1$：I/Oレディ

CPUと内部DMAに対するウェイト要求信号です．CPUに対してI/Oデバイスのスピードがついてこないときに，“L”レベルにすることでCPUにウェイトがかかり，I/Oアクセス時間を引き延ばします．通常は，CPUの速度に応じたウェイトが自動的に挿入されます．IORDY信号の“L”レベルの信号幅は最大7 μs以下にします．

〈図 3-4 (b)〉 バス・スロットのドライブ能力(286以降のCPU機)

信号名	1スロット当たりの最大入力電流		外部ロジックがドライブするときの最小出力電流(注)	
	I_{IL} (mA)	I_{IH} (μA)	I_{OL} (mA)	I_{OH} (mA)
AB$_{001}$~AB$_{231}$	−0.8	40	12	−1.2
BHE$_0$				
DB$_{001}$~DB$_{151}$				
IOR$_0$		50		
IOW$_0$				
MRC$_0$				
NWC$_0$				
MWE$_0$		40		
RFSH$_0$			不可	
IR$_{31}$~IR$_{131}$	—	—	8	−0.4
IOCHK$_0$	—	—		
INT$_0$	−0.8	40	不可	
NMI$_0$			—	—
SCLK$_1$			不可	
S18CLK$_1$				
POWER$_0$				
RESET$_0$				
DRQ$_{00}$, DRQ$_{30}$	—	—	8	−0.4
DACK$_{00}$, DACK$_{30}$	−0.8	40	不可	
WORD$_0$			8	−0.4
DMATC$_0$	−0.4	20	不可	
CPUENB$_{10}$	−0.8	40	12	−1.2
IORDY$_1$	—	—	8	−0.4
EXHRQ$_{10}$, EXHRQ$_{20}$	—	—		
EXHLA$_{10}$, EXHLA$_{20}$	−0.8	40	不可	
SBUSRQ$_1$				
NOWAIT$_0$	—	—	8	−0.4
SALE$_1$	−0.8	50	12	−1.2
INTA$_0$		40	8	−0.4
MACS$_0$	—	—		

注：外部ロジックは，IR$_{91}$，IOCHK$_0$，DRQ$_{00}$，DRQ$_{30}$，EXHRQ$_{10}$，EXHRQ$_{20}$，NOWAIT$_0$，MACS$_0$がオープン・コレクタで，他はトライ・ステート出力であること

▶ GND：グラウンド

▶ +5 V：+5 V電源ライン

▶ +12 V：+12 V電源ライン

▶ V$_1$，V$_2$：オプション電源ライン

オプション用の電源ラインは，本体側からは電源は供給されていません．また，すべての拡張スロットに接続されているので，他の拡張基板がV$_1$，V$_2$を使用していると，電源ラインがぶつかってしまう可能性があります．

以下の信号は，8086/V30用のバス専用の信号で，

〈図3-5〉1スロット当たりの電源容量

● PC9801/E/F/M/U/VF/VM/UV

DC	変動率	1スロット当たりの容量
＋ 5 V	± 5 %以内	0.5 A
＋12 V	±10 %以内	0.06 A
−12 V	±10 %以内	0.07 A

● 前記以外

DC	変動率	1スロット当たりの容量
＋ 5 V	± 5 %以内	0.8 A
＋12 V	±10 %以内	0.06 A
−12 V	±10 %以内	0.07 A

80286以降のCPUを使用したバスにはありません．また，PC9801LV21に，PC9801LV-08(I/O拡張ユニット)を使用したときは無効です．

▶ $DMAHLD_0$：DMAホールド

内部のDMAの動作をすべてイン・アクティブにし，内部DMAが動作しないようにする信号です．外部DMA等を使用するときに使います．この信号はDRAMのリフレッシュを止めますので，長時間(140クロック以上)アクティブにしてはいけません．

▶ HRQ_{00}：ホールド・リクエスト信号

CPUにホールドを要求する信号です．CPUはホールド要求されるとウェイト状態になります．

▶ $HLDA_{00}$：ホールド・アクノリッジ信号

CPUはホールド状態になったことを示す信号です．

▶ S_{00}，S_{10}，S_{20}：CPUステータス信号

CPUのマキシマム・モードにおける，S_0，S_1，S_2のステータス信号がそのまま出力されています．

▶ $RQGT_0$：リクエスト/グラウンド信号

CPUのマキシマム・モードにおける，RQ/GT信号です．

▶ $LOCK_0$：ロック信号

CPUのマキシマム・モードにおける，LOCK信号です．

▶ $CPKILL_0$

内部CPUをバスから切り離すための信号です．

● 80286以降の機種における信号線

以下の信号は，80286以降のCPUを使用したバス用の信号です．8086/V30用バス専用の信号と入れ替わりで導入されました．

▶ $EXHRQ_{10}$，$EXHRQ_{20}$：ホールド・リクエスト信号

外部CPU/DMAからのバス要求信号です．

▶ $EXHLA_{10}$，$EXHLA_{20}$：ホールド・アクノリッジ信号

外部CPU/DMAへのアクノリッジです．

▶ $SUBSRQ_1$：バス解放要求信号

内部DMA，リフレッシュ制御回路から外部CPU/DMAに対するバス解放要求信号です．

▶ $NOWAIT_0$：ノーウェイト信号

メモリを0ウェイトで動かすときの要求信号です．

▶ $SALE_1$：上位アドレス・バス・ラッチ信号

アドレス・バスAB_{171}～AB_{231}のラッチ要求信号です．

▶ $INTA_0$：外部CPUデータ要求信号

外部CPUから，8259(割り込みコントローラ)に対するデータ要求信号です．

▶ $MACS_0$：メモリ・ボード自己アクセス信号

オプションのメモリ・ボードが，自身に対してアクセスされていることを示すために出力する信号です．I/O拡張ユニットで使用し，オープン・コレクタ出力とします．

この信号を定義していないメモリ・ボードのI/O拡張ユニットでの動作は保証されません．

拡張スロットの電気的仕様

◈ 各信号線のドライブ能力

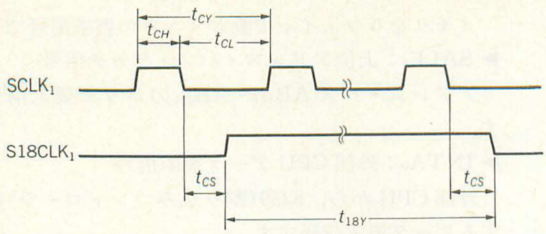
拡張スロットは，通常2～4個ありますが，例外的なものを除いて，バスはすべて並列に接続されています．拡張バスの入出力は，74F245(旧機種では74LS245)等のバス・バッファTTLでドライブされており，2～4個の拡張基板を同時にドライブすることになります．

LS-TTLのドライブ能力(ファンアウト)は20個ほどですが，これはDC(直流)的な計算で，AC(交流)的にはもっと少なくなります．そのため，規定された拡張スロット一つ当たりの入力電流は決して多くありません．アドレス/データ・バスや，一般的なコントロール信号では，1スロット当たりに許されている最大入力電流は−0.8 mA(I_{IL})ですから，LS-TTLで換算すると2個までということになります．

つまり，1枚の拡張基板で，1本のバス/制御線に二つの入力までつなぐことができます．また，TTLによっては，一つの入力で二つぶんのファンインを持つものもありますので注意が必要です．

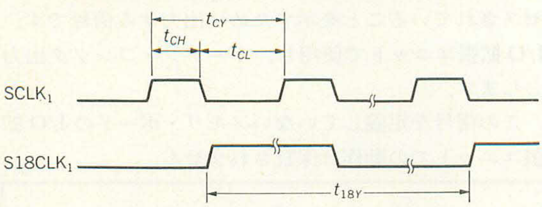
拡張基板の出力は，本体内部のバスをドライブするだけでなく，他の拡張基板の入力をもドライブすることになります．このため，拡張基板のドライブ電流は，最低12 mA(I_{OL})必要と既定されています．これは，標準的なLS-TTLでは満たすことができません．バッファ・タイプのLS-TTL(74LS245等)を使用する必要があります．74HC245等のCMOSバッファ(I_{OL}＝8 mA)や，8255等のLSIでも直接ドライブすることはできません．ただし，バスの制御信号には(I_{OL})は8 mAで良いものもあるので，これらは標準的なLS-TTL等でもドライブ可能です．

〈図 3-6〉 8086 のシステム・クロック



記号	パラメータ	8 MHz モード(ns)		5 MHz モード(ns)	
t_{CY}	SCLK Cycle Time	125.20		203.45	
記号	パラメータ	min(ns)	max(ns)	min(ns)	max(ns)
t_{CH}	SCLK High Time	43	57	70	84
t_{CL}	SCLK Low Time	68	82	120	134
t_{18Y}	S18CLK Cycle Time	307.200(kHz)		307.200(kHz)	
t_{CS}	S18CLK Delay Time	83	133	19	67

〈図 3-7〉 V30 以降の CPU のシステム・クロック



記号	486/Pentium*[1]		80286/386*[2]				70116			
			8/16 MHz		10/12/20 MHz		8MHz		10 MHz	
	min	max	min	max	min	max	min	max	min	max
t_{CY}	101.73		125.20		101.73		125.20		101.73	
T_{CH}	45	56	58	68	45(46)	56(55)	46	69	38	56
T_{CL}	45	56	58	68	45(46)	56(55)	56	79	46	64
t_{18y}	307.20(kHz)									

＊1：PC9801FA は 80386 の 16 MHz と同じ値をとる

＊2：PC9801US は 10/12/20 MHz の値をとる．( )内は PC-98XL 80386 の 16 MHz 時は 8 MHz モードと同等のクロックが，80386 の 20 MHz 時は，10 MHz モードと同等のクロックが，80286 の 12 MHz 時は 10 MHz モードと同等のクロックが出力される

IOCHK$_0$や IORDY$_1$等の制御線は，バス上で並列に，ワイヤード OR されている可能性がありますので，これらの制御線ではオープン・コレクタの IC でドライブする必要があります．

バス・スロットのドライブ能力を図 3-4 に示します．

◈ 電源容量

拡張スロットには，電源として，+5 V，+12 V，−12 V の 3 種類の電源が供給されています．拡張基板 1 枚当たりに許されている電源容量は図 3-5 のようになります．比較的電流をたくさん使用する基板等には辛い規格のようで，基板外に電源を用意して別に供給する拡張基板も多いようです．

◈ 信号線タイミング

▶システム・クロック(SCLK$_1$)

システム・クロックの周波数は，機種や CPU のクロック切り替えスイッチの位置によっても変わります．また，CPU が 8086 とそれ以外の CPU とでは，クロックのデューティ比が違います．

8086 のシステム・クロックを図 3-6 に，V30 以降の CPU のシステム・クロックを図 3-7 に示します．

8086/V30 用バスと 80286 以降の CPU 用バスでは，タイミングが変更されているために，8086/V30 専用に設計されたメモリ・ボードは動かない可能性があります．両者の違いは以下のとおりになっています．

① アドレス・バスの上位 7 ビットは本体側でラッチされていないため，メモリ側でアドレス・デコードした後，SALE$_1$信号によりラッチする．また，自分のアドレスであった場合は，MACS$_0$信号を駆動する(MACS$_0$は拡張ユニットで使用)．

② メモリ・アクセスを 0 ウェイト(バス・サイクル 2 クロック)で動作させる場合は，NOWAIT$_0$信号を駆動すること．

③ DRAM に対するアクセスを行う場合は，ディレイド・ライト・サイクル・モードを使用すること．

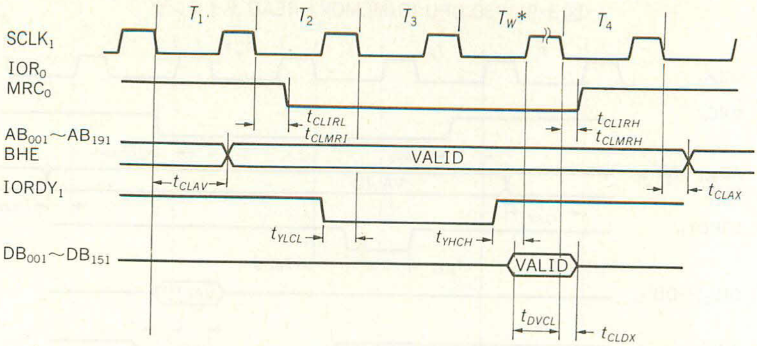
最近では，拡張メモリ・ボードの製作は，市販品よりコストもかかり自作する意味が薄くなってきました．

以下の 80286 以降の CPU でのアクセス・サイクルの解説は，拡張領域でのメモリ(主に ROM)アクセス・サイクルに限定して書いてあります．この領域でのアクセスは，自動的にウェイトが挿入されます．

▶メモリ・リード・サイクル

メモリ・リード・サイクルは，8086/V30 では 4 ステート，80286 以降の CPU では 2 ステートと異なっています．しかし，メモリ・リードでメモリに求められるアクセス速度は，MRC$_0$がアクティブになってから，

〈図 3-8〉
8086 CPU I/O
MEMORY READ
タイミング



＊I/O リード　5MHz-1wait　8MHz-2wait
メモリ・リード　5MHz-No wait　8MHz-1wait

CPU		8086-5 M	
記　号	パラメータ	min (ns)	max (ns)
t_{CLRL}	Command Active Delay	0	35
t_{CLRH}	Command Inactive Delay	0	35
t_{CLAV}	ADDRESS Valid Delay		128
t_{CLAX}	ADDRESS Hold Time	10	
t_{YLCL}	IORDY Inactive Setup	87	
t_{YHCH}	IORDY Active Setup	102	
t_{DVCL}	Read DATA Setup Time	54	
t_{CLDX}	Read DATA Hold Time	10	

CPU		8086			
記　号	パラメータ	8 MHz モード		5 MHz モード	
		min (ns)	max (ns)	min (ns)	max (ns)
t_{CLMRL}	MRD Active Delay	0	35	0	35
t_{CLMRH}	MRD Inactive Delay	0	35	0	35
t_{CLAV}	ADDRESS Valid Delay		78		128
t_{CLAX}	ADDRESS Hold Time	0		0	
t_{DVCL}	Read DATA Setup Time	40		54	
t_{CLDX}	Read DATA Hold Time	10		10	
t_{MRLH}	MRD Pulse Width	376 (3 T)		407 (2 T)	
t_{IRLH}	IOR Pulse Width	501 (4 T)		610 (3 T)	
t_{CLIRL}	IOR Active Delay	80	138	14	74
t_{CLIRH}	IOR Inactive Delay	14	74	14	74
t_{YLCL}	IORDY Inactive Setup	159		237	
t_{YHCH}	IORDY Active Setup	116		168	

読み出し CPU がデータ読み取りを行うまでのタイミングは基本的には似ています．

メモリに求められる速度は，8086/V30 の場合，

$$T_{acc}=(2+n)\times T_{cy}-T_{CLMRL}-T_{DVCL}$$

（n：CLOCK，5 MHz＝0，8 MHz＝1，10 MHz＝1）

で算出でき，80286 以降の CPU では，t_{mRDS}で規定されます．

▶ I/O リード・サイクル

I/O リード・サイクルは，8086 と V30 で，IOR_0のアクティブになるタイミングが違います．もちろん 80286 以降の CPU も，メモリ・リード・サイクル同様にステート数が違います．

I/O デバイスに求められる速度は，8086 の場合はメモリ・リード・サイクルと同等な算出方法で，

$$T_{acc}=(2+n)\times T_{cy}-T_{CLIRL}-T_{DVCL}$$

（n：CLOCK，5 MHz＝1，8 MHz＝2）

で算出でき，V30 の場合は，

$$T_{acc}=(1+n)\times T_{cy}-T_{CLIRL}-T_{DVCL}$$

（n：CLOCK，8 MHz＝2，10 MHz＝3）

となります．80286 以降の CPU では，T_{IORDS}で規定されます．

▶ メモリ・ライト・サイクル

メモリ・ライト・サイクルは，メモリ・リード・サイクルと同様なタイミングです．データ・バスの内容が確定したことを知るためには，MWE_0を使用します．

メモリに求められる速度は，8086/V30 の場合，

$$T_{acc}=(1+n)\times T_{cy}-T_{CLMWL}+T_{CLWH}$$

（n：CLOCK，5 MHz＝0，8 MHz＝1，10 MHz＝1）

で算出できます．

▶ I/O・ライト・サイクル

I/O ライト・サイクルは，メモリ・ライト・サイクルにくらべて IOW_0がアクティブになるタイミングが遅れます．V30 の I/O リード・サイクルに似ています．

I/O デバイスに求められる速度は，8086/V30 の場合，

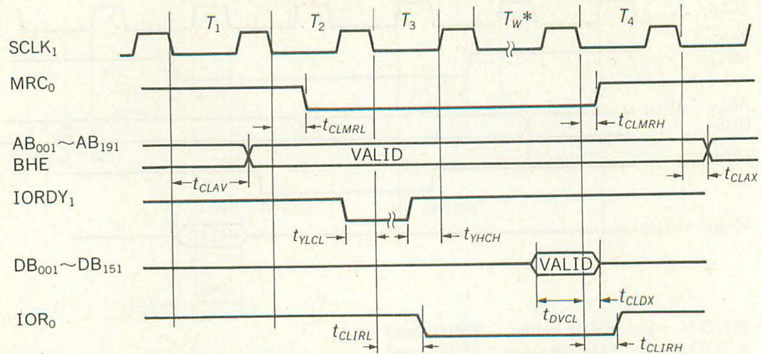
$$T_{acc}=(1+n)\times T_{cy}-T_{CLIL}+T_{CLIH}$$

（n：CLOCK，5 MHz＝1，8 MHz＝2，10 MHz＝3）

で算出できます．

図 3-8 に 8086，図 3-9 に V30 の CPU　IO/MEM

〈図 3-9〉 V30 CPU IO/MEMORY READ タイミング



＊I/O リード　8MHz-2wait　10MHz-3wait
メモリ・リード　8MHz-1wait　10MHz-1wait

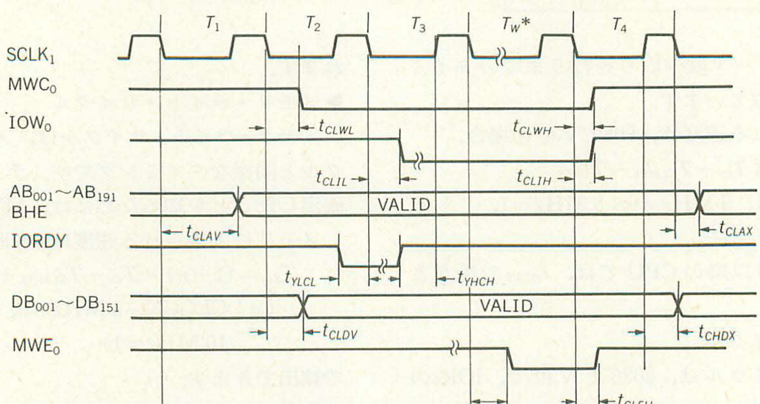
CPU		70116 (V30)			
		8 MHz モード		10 MHz モード	
記号	パラメータ	min (ns)	max (ns)	min (ns)	max (ns)
t_{CLMRL}	MRD Active Delay	−45	38	−45	38
t_{CLMRH}	MRD Inactive Delay	0	35	0	35
t_{CLAV}	ADDRESS Valid Delay		78		68
t_{CLAX}	ADDRESS Hold Time	0		0	
t_{DVCL}	Read DATA Setup Time	38		28	
t_{CLDX}	Read DATA Hold Time	10		10	
t_{MRLH}(*)	MRD Pulse Width	376 (3*T*)		305 (3*T*)(***)	
t_{IRLH}(*)	IOR Pulse Width	361 (4*T*)		392 (5*T*)	
t_{CLIRL}	IOR Active Delay	−47	50	−47	50
t_{CLIRH}	IOR Inactive Delay	0	35	0	35
t_{YLCL}(**)	IORDY Inactive Setup				
t_{YHCH}	IORDY Active Setup	60		43	

＊：CPU 外部から Wait をかけないとき

＊＊：IORDY はバスに対して非同期でよい．本規格値内でクロックに対して変化させれば，次の Cycle の動作が保証される

＊＊＊：オプション ROM のアドレス空間では 407 (4*T*) となる

〈図 3-10〉 8086/V30 CPU IO/MEMORY WRITE タイミング

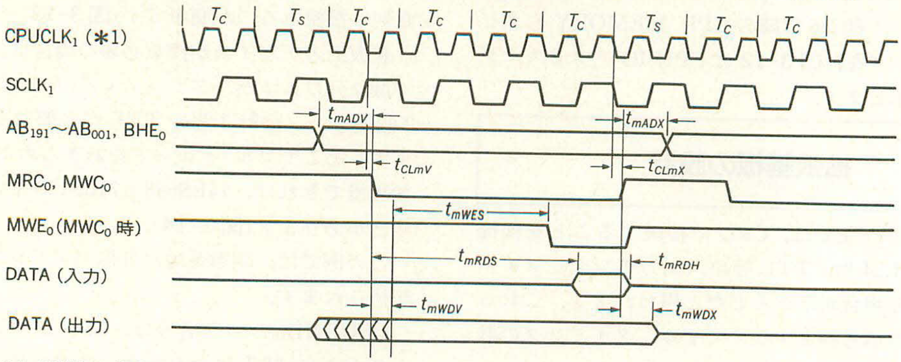


＊I/O ライト　5MHz-1wait　8MHz-2wait
10MHz-3wait
メモリ・ライト　5MHz-No wait　8MHz-1wait　10MHz-1wait

CPU		8086−5 MHz		8086				70116 (V30)			
CLOCK		5 MHz モード		8 MHz モード		5 MHz モード		8 MHz モード		10 MHz モード	
記号	パラメータ	min (ns)	max (ns)	min (ns)	max (ns)	min (ns)	max (ns)	min (ns)	max (ns)	min (ns)	max (ns)
t_{CLWL}	MWC Active Delay	0	35	0	35	0	35	−45	38	−45	38
t_{CLWH}	MWC Inactive Delay	0	35	0	35	0	35	0	35	0	35
t_{CLIL}	IOW Active Delay	−70	75	43	74	−70	74	14	100	14	100
t_{CLIH}	IOW Inactive Delay	15	75	14	74	14	74	0	35	0	35
t_{CLDV}	Write Data Valid Delay		122		72		122		78		68
t_{CHDX}	Write Data Hold Time	10		10		10		10		10	
t_{CLEL}	MWE Active Delay	114	211	11	37	11	37		69		41
t_{CLEH}	MWE Inactive Delay	40	130	19	65	19	65		120		34

〈図 3-11〉 80286 以降 CPU MEMORY アクセス・タイミング

(0C000H～0DFFFFH, FC0000H～FDFFFFH, 080000H～09FFFFH　RAM KILL 時)



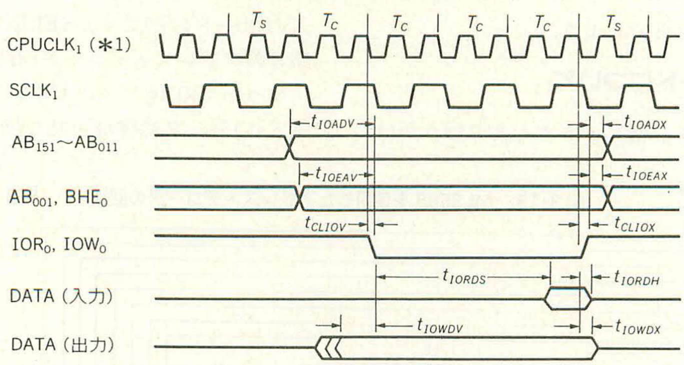
*1:CPUCLK1 信号は拡張バス上へは出力されない

記号	80286/386									
	8 MHz		10 MHz		12 MHz		16 MHz		20 MHz	
	min	max	min	max	min	max	min	max	min	max
t_{mADV}	50		50		40		50		50	
t_{mADX}	18		18		15		18		18	
t_{CLmV}	3	25	3	21			3	25	3	21
t_{CLmX}	3	25	3	20			3	25	3	20
t_{mWDS}	302		332		302		302		329	
t_{mRDS}		308		332		308		295		317
t_{mRDH}	5		5		5		5		5	
t_{mWDV}		−17		−55		−55		−17		−55
t_{mWDX}	15		14		14		15		14	

注意：PC9801US は 20 MHz の値をとる

記　　号		486/Pentium	
		min	max
t_{mADV}		50	
t_{mADX}		18	
t_{CLmV}	拡張バス上	3	21
	I/O 拡張ユニット上	24	67
t_{CLmX}	拡張バス上	3	20
	I/O 拡張ユニット上	12	45
t_{mWDS}		329	
t_{mRDS}			317
t_{mRDH}		5	
t_{mWDV}			−55
t_{mWDX}		14	1

〈図 3-12〉 80286 以降 CPU I/O アクセス・タイミング



*1:CPUCLK1 信号は拡張バス上へは出力されない

記号	8 MHz		10 MHz		12 MHz		16 MHz		20 MHz	
	min	max	min	max	min	max	min	max	min	max
t_{IOADV}	115		139(150)		139		115		136	
t_{IOEAV}	111		139(150)		139		111		136	
t_{IOADX}	25		21(32)		21		25		21	
t_{IOEAX}	25		21(32)		21		25		21	
t_{CLIOV}	3	25	3	21			3	25	3	21
t_{CLIOX}	3	25	3	20			3	25	3	20
t_{IORDS}		270		254(312)		254		257		239
t_{IORDH}	5		5		5		5		5	
t_{IOWDV}	67		140(151)		130		67		140	
t_{IOWDX}	23		21(32)		21		23		21	

注意：PC9801US は 20 MHz の値をとる．( )内は PC-98XL

記　　号		486/Pentium	
		min	max
t_{IOADV}		136	
t_{IOEAV}		136	
t_{IOADX}		21	
t_{IOEAX}		21	
t_{CLIOV}	拡張バス上	3	21
	I/O 拡張ユニット上	10	40
t_{CLIOX}	拡張バス上	3	20
	I/O 拡張ユニット上	10	39
t_{IORDS}			239
t_{IORDH}		5	
t_{IOWDV}		140	
t_{IOWDX}		21	

ORY READタイミングを，図3-10に8086/V30 CPU IO/MEMORY WRITEタイミングを示し，また，図3-11に80286以降のCPU MEMORYアクセス・タイミングを，図3-12にCPU IOアクセス・タイミングを示します．

拡張基板の設計

拡張基板の作成では，CPUに接続することを前提に作られたLSIや，TTL等の入出力インターフェースを使用する場合がほとんどだと思われます．これらは，使用する信号線も少なくてすみ，タイミングの計算なども比較的簡単にすみます．

特に80系のCPUへの接続を前提に作られたLSI(μPD8255等)は，極めて簡単に接続できるようになっています．

◈ I/Oアドレスについて

ユーザに開放されているI/Oポート・アドレスは，×nDOH～×nEFHで，nは0～7までです．

n=8～FまではNECのリザーブになっていますから，今後，拡張基板を設計する場合は最低でも12ビット，できれば16ビット(フル)デコードする必要があると思われます．

他の拡張基板でも，この領域を使用している基板は数多くありますので，それらとかち合わないためにも，I/Oアドレスは固定せずに，ディップ・スイッチ等で変更可能な設計にするべきでしょう．

◈ アドレス・デコードについて

アドレス・デコードを，何ビットぶんまで行うかは，コストや部品点数に関係してきます．16ビット・フル・デコードするなら，74LS688というTTLを2個直列に接続するのが簡単です(図3-13)．

基板上のデバイスが複数必要な場合は，74LS138等で振り分けますが，アドレス・デコードに74LS688を2個使うと，合計3個のTTLが必要になってしまいます．そこでフル・デコードをあきらめ12ビット程度で我慢できれば，74LS688+74LS138だけでも十分な場合があります(図3-14)．

この例では，偶数番地に8個のアドレス・デコードが得られます．

$CPUENB_{10}$という信号は，CPUがバスを使用しているときに“L”になります．アドレス・デコードの際に，いっしょにデコードさせておきます．

◈ ワード・アクセスとバイト・アクセス

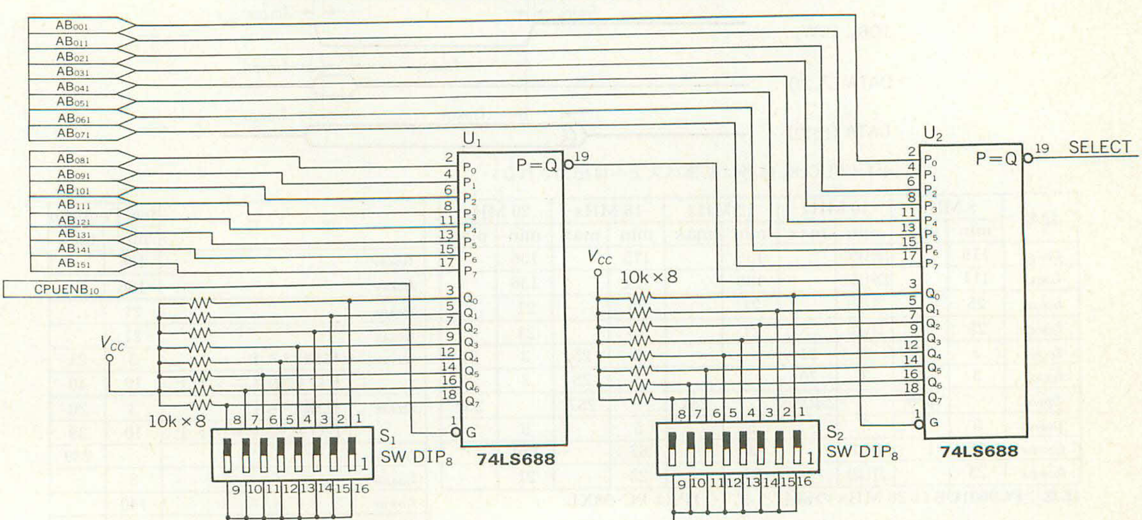
PC98シリーズでも数多く使用されている80系周辺LSIは，データ・バスの幅が8ビット(バイト)ですから，バイト・アクセスになります．8086等16ビットCPUは，偶数アドレスが下位8ビット，奇数アドレスが上位8ビットになりますから，これら8ビット・バスのLSIは奇数か，偶数のどちらかのアドレスに割り当てなくてはなりません．

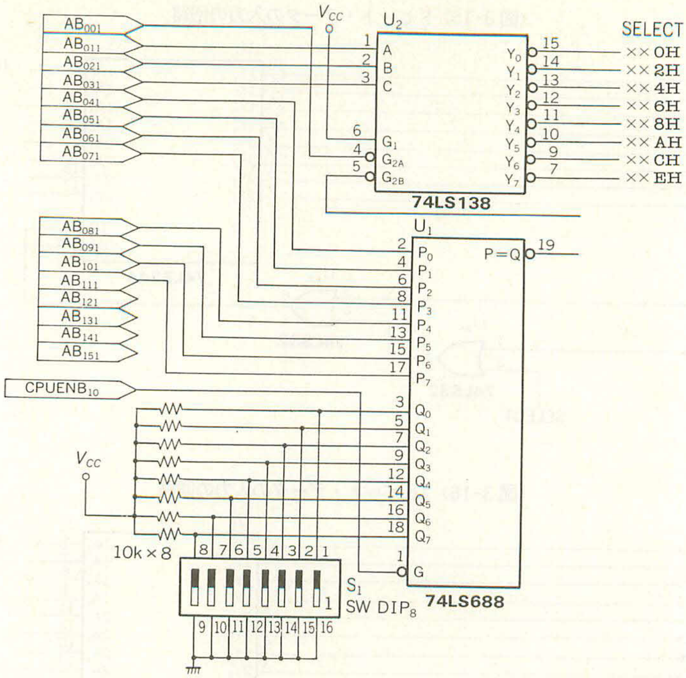
そのためアドレスのデコードは，偶数アドレスに配置する場合はAB_{001}を使用し，奇数アドレスの場合はBHE_0を使用します．

図3-15は，偶数アドレスに8ビット(1バイト)の入力ポートの例です．SELECTには，図3-14の回路等のアドレス・デコード回路を接続します．

せっかくの16ビットCPUですから，16ビット(ワード)のデータ処理が可能な場合，16ビットぶん一度

〈図3-13〉74LS688を使用したアドレス・デコーダの回路





〈図 3-14〉
74LS688+74LS138
を使用したアドレス・
デコーダの回路

に入出力できると効率も上がって効果的です．この場合は，8ビットのポートが奇数，偶数の二つにあるという考え方で，アドレス・デコードをします．具体的には図3-16のようなデコード方法が良いでしょう．

このような方法ですと，偶数バイトだけや奇数バイトだけのアクセス時でも，またワード・アクセス時でも，ちゃんとアクセスできます．74LS244の代わりに74LS374等を使用すれば出力ポートにもできます．

◈ バス・バッファについて

TTL等のICの出力の能力は有限です．拡張スロット等に使用されている74LS245等のTTLでは，出力が“L”(0 V)のときに，24 mAまでの電流に耐えられます．反対に，74LS00等の一般のTTLの入力を“L”にするためには，−0.4 mA流す必要があります．つまり，74LS245の出力には，最大60個までの74LS00の入力をつなげることができます．これをファンアウト(出力)/ファンイン(入力)と呼びます．これはDC(直流)的な計算で，AC(交流)的にはもっと少なくなります．

PC98シリーズの拡張スロットにも，ファンアウト/ファンインが規定されています．アドレス/データ・バスや，一般的なコントロール信号では，許されている最大入力電流は，最大−0.8 mAですから，74LS00等の標準的なTTLで換算すると2個までということになります．さらに，TTL(74LS266)によっては，一つの入力で二つぶんのファンインを持つものもありますので注意が必要です．

拡張基板のアドレス/データ・バスや，主要なコントロール信号の既定された出力電流は，最低12 mA必要とされています．これは，74LS00等の標準的なLS-TTLの出力では満たすことができません．74LS245等のバッファ・タイプのLS-TTLを使用する必要があります．

図3-15のように，極めて簡単なTTL入出力インターフェースならば，バッファは必要ありませんが，入出力のポートが増えたり，複雑なロジックが必要になると，バスに2個以上のICがぶら下がることになり，別にバス・バッファが必要になってきます．また，LSI等のMOS系のデバイスの場合も，一般的には出力電流が多く取れないのでバス・バッファが必要になります．

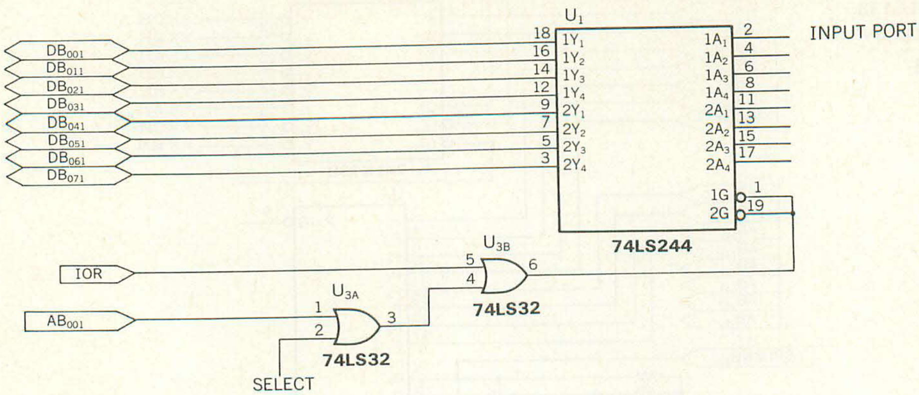
◈ 入力ポートの設計

入力ポートのアクセス・タイミングについて考えてみましょう．

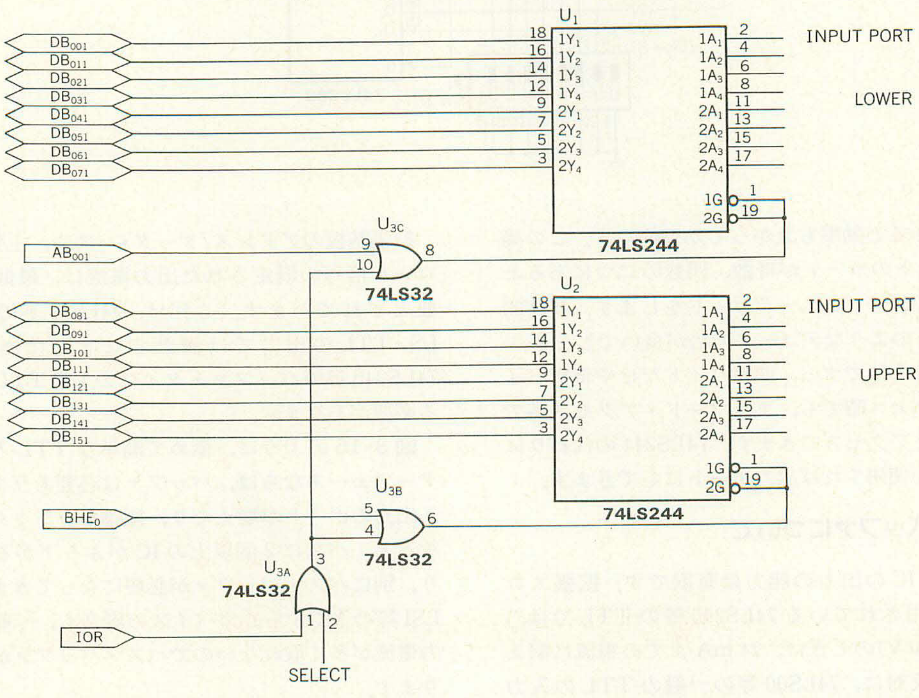
拡張基板のLSIやTTLからデータを読み込むときは，まずCPUが出力したI/Oアドレスが，自分のアドレスかどうか認識しなければなりません．そのときに使用するアドレス・デコーダは，アドレスが決定してから結果がわかるまで23 ns(74LS688)の時間がかかります．アドレスをフル・デコードして，アドレス・デコーダが2段になっていれば，2倍の46 nsかかる計算です．

次に，CPUの読み出し制御信号IOR_0が“L”になり，LSIやTTLが読み出しモードになります．実際

〈図 3-15〉 8 ビット・データの入力の回路



〈図 3-16〉 16 ビット・データの入力の回路



には，アドレスが確定してから 100 ns ほど経って IOR_0が“L”になるので，先ほどのアドレス・デコーダの延長時間(46 ns)は問題にはなりません．ただし，アドレス・デコードに複雑な回路を使い，100 ns を超えるようになると，タイミングを新たに考えなくてはなりませんので注意が必要です．

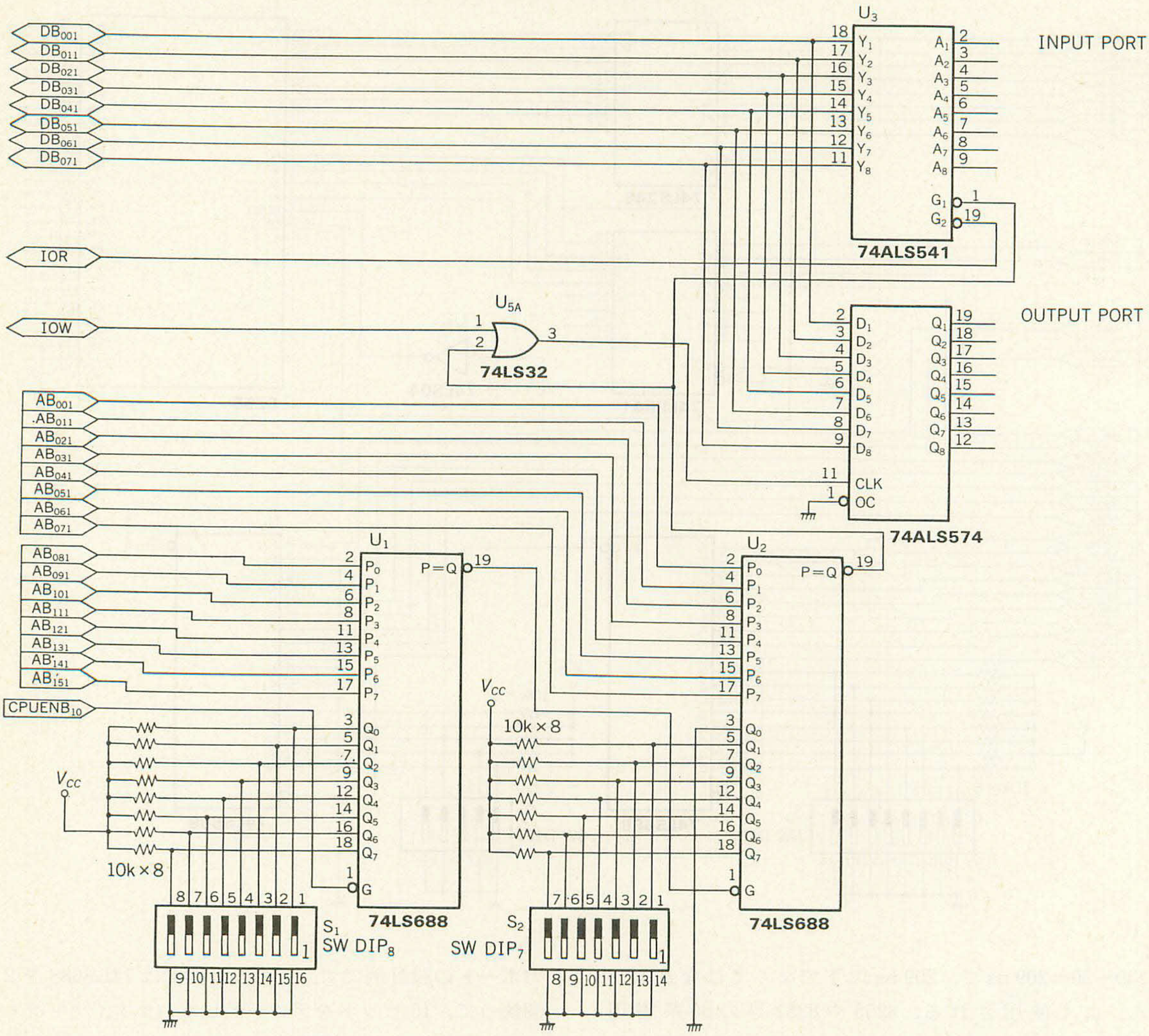
IOR_0信号が“L”になると，アクセスされた LSI や TTL から CPU に向けてデータを出します．TTL 等では 30 ns(74LS541)程度でデータが出てきますが，LSI では 100 ns 以上かかるのが普通です．その種類によってデータが出てくる時間が違うので，使用する LSI の仕様を調べておく必要があります．

IOR_0が“L”になってから，CPU が実際にデータを読み出すまでの時間は規定されています．最も速いタイミングを要求されるのは，CPU のクロックが 20 MHz のときで，239 ns となっています．前記のとおり TTL では 30 ns なので問題ありませんが，LSI では 239 ns より速くなくてはいけません．

さらに問題になるのは，バス・バッファを付けた場合です．一般的なバス・バッファ(74LS245)では，データがバッファを通り抜けるのに 12 ns ほどの延長時間があります．また，IOR_0信号にもバッファ(74LS244)が入っていると 18 ns ほど延長時間があるので，合わせて 30 ns ほど無駄に時間が浪費されてしまいます．

これから計算すると，LSI のアクセス・スピードは，

〈図 3-17〉TTL を使用した入出力ポート



サンプル・プログラム・ディスク頒布のご案内

本書に紹介されたサンプル・プログラムのソース・ファイルおよび実行用バイナリ・ファイルをディスク頒布します.

● **頒布価格**　3,000 円（送料，税込み）

● **頒布メディア**　3.5 インチまたは 5 インチ 2HD
（記入のない場合は3.5インチをお送りします）

● **申し込み期限**　1996 年 12 月末日

● **申し込み方法**

下記の申し込み用紙（コピー可）に必要事項を記入のうえ，代金を同封し**現金書留**でお申し込みください.

● **申し込み先**

〒170　東京都豊島区巣鴨 1-14-2
CQ 出版㈱
ディスク・サービス TRSP45 係

● **トランジスタ技術 SPECIAL No. 45 ソフトウェア申し込み用紙**　TRSP45

送り先ご住所：〒

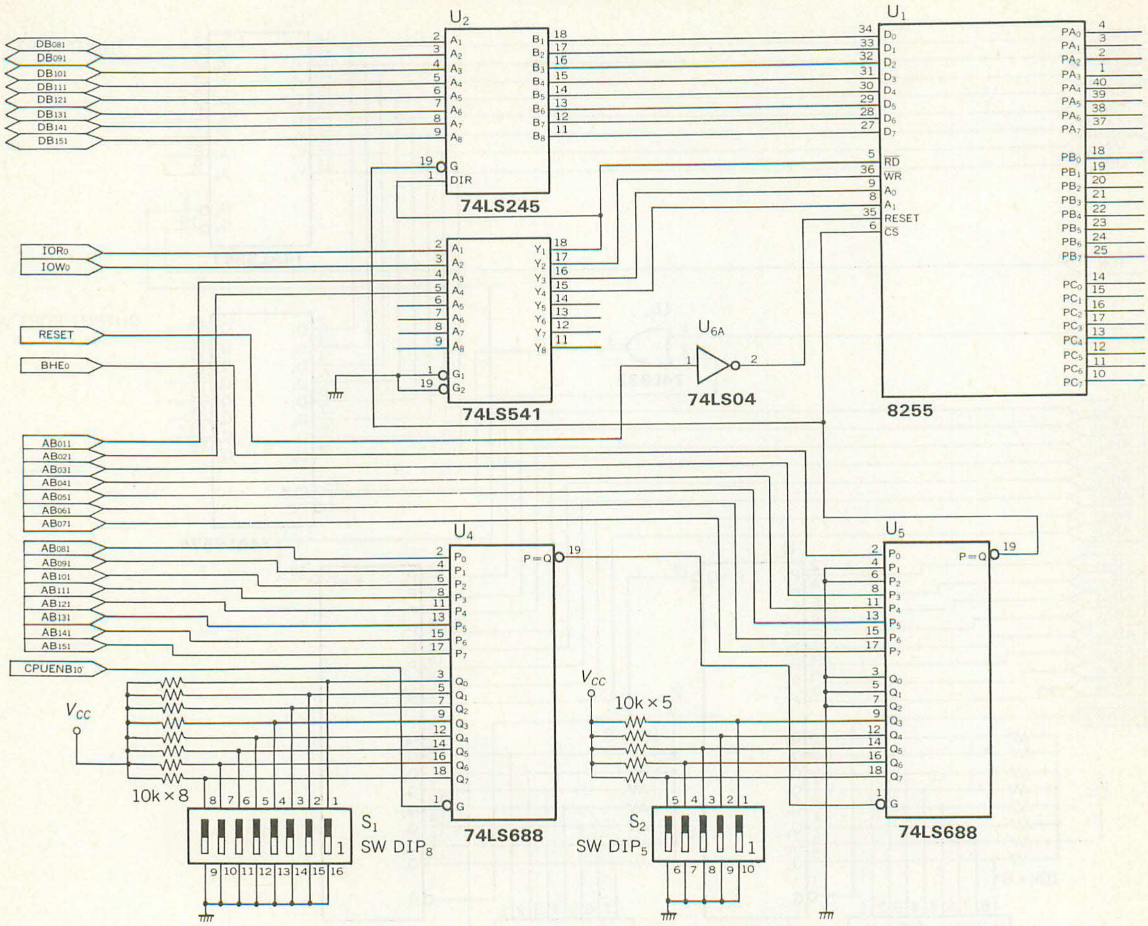
お名前：　　　　☎（　　）

▶希望メディア（✓を入れてください）

☐ 5 インチ 2HD

☐ 3.5 インチ 2HD

〈図 3-18〉 μPD71055 を使用した入出力ポート



239－30＝209 ns で，209 ns 以下でなくてはなりません．よく使用される，8255 や 8251 等の 80 系周辺 LSI では，アクセス・スピードが 150〜300 ns ほどなので，μPD8251AC-2 や，μPD71055 等の高速タイプ (160 ns) を使用しなくてはなりません．

◈ 出力ポートの設計

一般に，出力ポートのアクセス・タイミングは，入力ポートに比べて長めです．

出力の場合は，IOW_0信号が“L”になってから書き込みモードに入り，“H”になったときにデータを取り込みますので，IOW_0信号の長さが，アクセス・スピードになります．タイミング的にはIOR_0より長くなります．

また，書き込まれるデータは，IOW_0が“L”になる前に確定していますから，多少のバス・バッファの延長には関係ありません．

◈ TTL の入出力ポートの設計

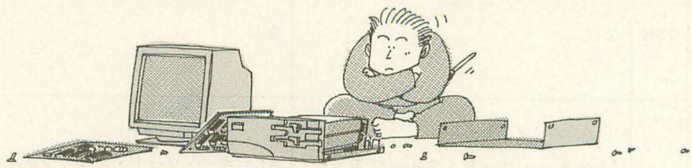
図 3-17 は，TTL だけで構成した 8 ビットの入出力ポートの設計例です．この設計例では 74LS688 を 2 個使って，15 ビットをデコードしています．データ・バスが下位バイトに接続されているので，偶数アドレスを指定しないと動作しませんので，最下位ビットのアドレス・デコードは固定してあります．

◈ 80 系周辺 LSI の接続

図 3-18 は，μPD71055(8255) を使用した入出力ポートの設計例です．この設計例では 74LS688 を 2 個使って，13 ビットをデコードしています．8255 が四つのアドレスを持つのと，データ・バスが下位バイトに接続されているので，偶数アドレスを指定しないと動作しませんので，下位の 3 ビット分は固定されています．

また，8255 等の 80 系周辺 LSI では，RESET 入力が正論理で，PC9801 シリーズの拡張スロットとは論理が反対になります．このために，74LS04 で反転する必要があります．

4 PC98シリーズのBIOS



PC9801シリーズが持つ周辺ハードウェアの制御のほとんどは，BIOSを操作することで使用できます．この BIOSは，本体内蔵のROMや拡張インターフェース基板上のROMでサポートされます．BIOSのアクセスにはソフトウェア割り込み(INT命令)が使用されていて，18H～1CHまでが使用されています．

18H　キーボード，CRT，グラフィック
19H　RS-232-C
1AH　プリンタ
1BH　DISK
1CH　カレンダ時計，タイマ

各BIOSの呼び出しは，AHレジスタに機能番号を入れて，対応するINT命令を実行することで，BIOSを振り分けています．このため，五つの割り込みだけでも多くの機能を処理できます．

また，BIOSへ受け渡すパラメータは，主にレジスタを使用しています．多量のパラメータを受け渡す場合には，メモリ上にパラメータを置き，そのメモリのアドレスをレジスタへ入れて呼び出す場合もあります．また，BIOS処理で起こったエラーは，キャリ・フラグ(CF)の状態やAHレジスタの内容で知ることができます．

キーボードBIOS

キーボードからの割り込みを処理し，内部のバッファにキー・データを格納するモジュールと，プログラムからのキーボード入力要求を受けて，内部のバッファからキー・データを返すモジュールからできています．

キー・コードは，キーボードから送られてくるデータで，それぞれのキーに割り当てられたデータです．キー・データはキー・コードを元に作られたASCII(JIS)コードで，ユーザ・プログラムを扱うデータです．

割り込み番号 INT	機能番号 AH	入力パラメータ	戻り値	機能
18H	00H	なし	AH＝キー・コード AL＝キー・データ	「キー・データの読み出し」 キー・バッファ先頭に格納されているキー情報をキー・データ・コードに変換して読み出す． キー・バッファが空であれば，なにか入力されるまで待つ． 読み終わったキー情報はキー・バッファから失われる．
	01H	なし	AH＝キー・コード AL＝キー・データ BH＝AXに読み出したデータの状態 00H＝無効 01H＝有効	「キー・バッファ状態のセンス」 戻り値のAH，ALは「キー・データの読み出し」と同様であるが，キー・バッファの状態は変わらない．キー・バッファが空であれば「無効」を返す． 「キー・バッファ状態のセンス」でキー・データが有効であるのを確認したうえ，「キー・データの読み出し」を実行すれば，キー・バッファが空であっても，BIOS内部でキー入力待ちすることはない．

INT	AH	入力パラメータ	戻り値	機　能
18H	02H	なし	AL＝シフト・キーの状態	「シフト・キー状態のセンス」 現在押されているシフト・キーの状態を調べる．
	03H	なし	なし	「キーボード・インターフェースの初期化」 キーボード・インターフェースで使用している μPD8251 の初期化と，BIOS で使用しているメモリ・エリアの初期化を行う．
	04H	AL＝キー・コード・グループ番号 (00H～0FH)	AL＝キー・コード・グループ内の八つのキー状態	「キー入力状態のセンス」 キーボード上のキーを 16 個のグループに分けて，そのグループのそれぞれのキーが押されているか，離されているかを調べる． (キー・コード・グループは図 4-1 を参照)

〈図 4-1〉キー・コード・グループ

キー・コード・グループ ＼ ビット	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	ESC	1 ! ヌ	2 ” フ	3 # ア ァ	4 $ ウ ゥ	5 % エ ェ	6 & オ ォ	7 ’ ヤ ャ
1	8 (ユ ュ	9) ヨ ョ	0 ワ ヲ	— ＝ ホ	∧ ` ヘ	¥ ¦ —	BS	TAB
2	Q タ	W テ	ヨ イ ィ	R ス	T カ	Y ン	U ナ	I ニ
3	O ラ	P セ	@ ~ ゛	[{ 「 ゜	⏎	A チ	S ト	D シ
4	F ハ	G キ	H ワ	J マ	K ノ	L リ	; + レ	: * ケ
5	] } ム 」	Z ツ ッ	X サ	C ソ	V ヒ	B コ	N ミ	M モ
6	, < ネ 、	. > ル 。	／ ? メ ・	— ロ	SPACE	XFER	ROLL UP	ROLL DOWN
7	INS	DEL	↑	←	→	↓	HOME CLR	HELP
8	—	／	7	8	9	*	4	5
9	6	＋	1	2	3	＝	0	,
A	・	NFER	vf・1	vf・2	vf・3	vf・4	vf・5	
B							HOME	
C	STOP	COPY	f・1	f・2	f・3	f・4	f・5	f・6
D	f・7	f・8	f・9	f・10				
E	SHIFT	CAPS	カナ	GRPH	CTRL			
F								

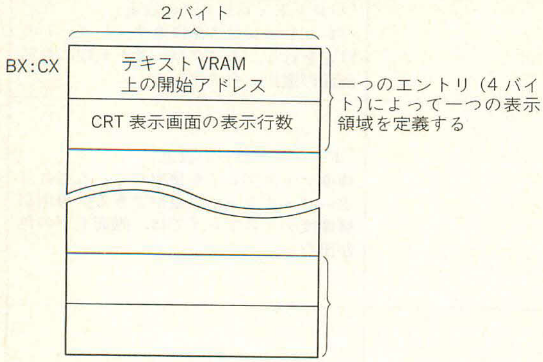
CRT BIOS

CRT 関連の制御を行います．画面表示等の開始・停止や，設定・初期化等をできますが，文字出力・グラフィックの描画はできません．また，ブザーの制御もできます．

割り込み番号 INT	機能番号 AH	入力パラメータ	戻り値	機能
18H	0AH	AL＝モード設定情報 D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 0 0 0 0 D_0= 画面当たりの行数 "0":25 行, "1":20 行 D_1= 行当たりの桁数 "0":80 字, "1":40 字 D_2=アトリビュート "0": バーチカル・ライン有効 "1": 簡易グラフ有効 D_3=KCG アクセス＊ "0": コード・アクセス有効 "1": ドット・アクセス有効 ＊: PC9801 では無効	なし	「CRT モードの設定」 テキスト表示用の GDC(μPD7220) 等のモード設定を行う．
	0BH	AL＝なし	AL＝モード設定状態 D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 D_0= 画面当たりの行数 "1":20 行, "0":25 行 D_1= 行当たりの文字数 "1":40 字, "0":80 字 D_2=アトリビュート・タイプ "1": 簡易グラフ, "0": バーチカル・ライン D_3=KCG アクセス・タイプ＊ "1": ドット・アクセス, "0": コード・アクセス D_7=CRT の種類 "1": 専用高解像度ディスプレイ "0": 標準ディスプレイ ＊: PC9801 では無効	「CRT モードのセンス」 「CRT モードの設定」で設定した情報を読み出す．
	0CH	AL＝なし	なし	「テキスト画面の表示開始」 テキスト画面表示用 GDC に表示開始要求を行う．
	0DH	AL＝なし	なし	「テキスト画面の表示停止」 テキスト画面表示用 GDC に表示停止要求を行う．
	0EH	DX＝表示する領域の開始アドレス (下位 16 ビット指定する)	なし	「一つの表示領域の設定」 テキスト画面全体を一つの表示領域に設定する．
	0FH	BX＝表示領域リストのセグメント・アドレス CX＝表示領域リストのオフセット・アドレス DH＝表示領域リストで最初に定義するエントリの表示領域番号(1〜3) 例えば，前回 3 分割しておき，今回 2 番目の表示領域を変更する場合，DH＝1 とする DL＝表示領域リストで定義するエントリ個数(1〜4) (図 4-2 参照)	なし	「複数の表示領域の設定」 テキスト画面を 4 まで分割し，それぞれの表示領域に設定する．
	10H	AL＝カーソルを点滅にするか否かの設定 00H＝点滅する 01H＝点滅しない	なし	「カーソル・タイプの設定」 カーソルを点滅にするか否かを設定する．
	11H	なし	なし	「カーソルの表示開始」 カーソルを表示する．

INT	AH	入力パラメータ	戻り値	機能
18H	12H	なし	なし	「カーソルの表示停止」 カーソルを表示しない．
	13H	DX＝カーソルを表示する位置の VRAMアドレス	なし	「カーソル位置の設定」 カーソルを表示する位置を設定する．
	14H	BX：CX＝フォント・パターン・バッファの先頭アドレス ANKコードの場合 DL＝展開するコード DH＝CRTタイプ (00H＝標準CRT用 80H＝専用高解像度CRT用) 漢字コードの場合 DX＝展開するコード 下位アドレス　上位アドレス ［制御域 2バイト］［フォント・パターン・バッファ (8, 16, 32バイトのいずれか)］ ↑BX:CX(フォント・パターン・バッファの先頭アドレス)	なし	「フォント・パターンの読み出し(16ドット)」 設定されたフォントのパターンを，フォント・パターン・バッファへ読み出す．
	15H	なし	AH＝ライト・ペンの状態 00H＝押されている 01H＝押されていない DX＝ライト・ペンが押された位置に対応するテキストVRAMのアドレス	「ライト・ペン位置読み出し」 ライト・ペンが押されたかを通知する．押された場合はその位置を知らせる．
	16H	DH＝アトリビュートをクリアする文字 DL＝表示エリアをクリアする文字 D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 G R B VL UL RV BL $\overline{ST}$ D_0＝シークレット D_1＝ブリンキング D_2＝リバース D_3＝アンダ・ライン D_4＝バーチカル・ライン カラーCRT / モノクロCRT D_5＝ 青 1 0 0 D_6＝ 赤 1 0 0 3bitによる D_7＝ 緑 1 1 0 濃淡表示 明 中 暗	なし	「テキストVRAMの初期化」 テキストVRAMの全領域を指定された文字でクリアする．
	17H	なし	なし	「ブザーの起呼」 ブザーを鳴らす．
	18H	なし	なし	「ブザーの停止」 ブザーを停止．
	19H	なし	なし	「ライト・ペン押下状態の初期化」 ライト・ペンが押された状態を検出するための状態表示をクリアする．
	1AH	BX：CX＝フォント・パターン・バッファの先頭アドレス 下位アドレス(先頭)　上位アドレス ［2バイト・ワーク・エリア］［フォント・パターン 32バイト］ └(破壊される) DX＝登録コード	なし	「ユーザ文字の定義(16ドット)」 ユーザの作成した文字，記号のフォント・パターンをKCG RAMへ登録する．登録された文字，記号は漢字と同様に，テキストVRAMに登録コードを書き込むだけで表示される．
	1BH	AL＝KCGアクセス・モードの設定 00H＝コード・アクセス 01H＝ドット・アクセス	なし	「KCGアクセス・モードの設定」 KCGアクセス・モードを，コード・アクセスまたは，ドット・アクセスに設定する．ドット・アクセスを選択すると，漢字，ユーザ定義文字の表示ができなくなる．

〈図 4-2〉表示領域リストの形式



テキスト VRAM 上の開始アドレス（2バイト）：
GDC からみたアドレスを指定
（0000H ～ 1000H－1）
CRT 表示画面の表示行数（2バイト）：20行モードの場合（1～20）
25行モードの場合（1～25）

グラフィック BIOS

GDC の操作だけで可能な，比較的簡単なグラフィックの制御が可能です．拡張グラフィック機能(4096色中16色モード)等はサポートされていません．

グラフ BIOS 制御領域を図 4-3 に示します．

〈図 4-3〉グラフ BIOS 制御領域

オフセット	ラベル	機能	サイズ
0000	GBON_PTN	3画面同時書き込み時の描画画面ナンバと描画オペレーション・モードの指定	RW_1
0001	GBBCC	セットするボーダ・カラー・コード	RB_1
0002	GBDOTU	単一画面の描画オペレーション・モードの指定	RB_1
0003	GBDSP	描画開始方向	RB_1
0004	GBCPC	パレット・レジスタにセットするカラー・コード	RB_4
0008	GBSX1	描画開始アドレス X 座標	RW_1
000A	GBSY1	描画開始アドレス Y 座標	RW_1
000C	GBLNG1	書き込み長さ／第1描画方向ドット数	RW_1
000E	GBWDPA	描画パターン・バッファの開始アドレス	RW_1
0010	GBRBUF		RW_3
0010	GBRBUF1	読み出しバッファ1の開始アドレス	
0012	GBRBUF2	読み出しバッファ2の開始アドレス	
0014	GBRBUF3	読み出しバッファ3の開始アドレス	
0016	GBSX2	描画終了アドレス X 座標	RW_1
0018	GBSY2	描画終了アドレス Y 座標	RW_1
001A	GBMDOT	マスキング・ドット数	RW_1
001C	GBCIR	半径	RW_1
001E	GBLNG2	第2描画方向ドット数	RW_1
0020	GBLPTN	線種パターン	RW_1
0020	GBMDOTI	基本パターン情報	RW_8
0028	GBDTYP	描画タイプ	RW_1
0029	GBFILL		RW_1
002A-48	ワーク・エリア		

割り込み番号 INT	機能番号 AH	入力パラメータ	戻り値	機能
18H	40H	なし	なし	「グラフィック画面の表示開始」 グラフィック画面を表示する．
	41H	なし	なし	「グラフィック画面の表示停止」 グラフィック画面を消す．
	42H	CH＝CRT ディスプレイのモード指定，VRAM 領域の指定(図 4-4 参照) D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 (—, —, —, —, 0, 0, 0, 0) D_4＝表示バンクの指定 "0"：バンク0を使用，"1"：バンク1を使用 D_5＝CRT ディスプレイのモード指定 "0"：カラー，"1"：モノクロ D_6, D_7＝VRAM 領域指定 "01"：UPPER，"10"：LOWER，"11"：ALL 注：PC9801/U2 では，表示バンクの指定は不可，"0" としておくこと	なし	「表示領域の設定」 表示対象とする描画メモリ領域を設定する．CRT ディスプレイのモード(モノクロ/カラー)と，使用する VRAM の領域を指定する(ALL＝400ライン/UPPER，LOWER＝200ライン)．

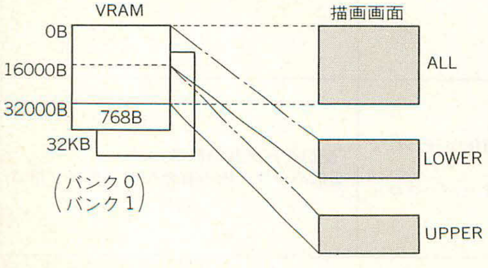
INT	AH	入力パラメータ	戻り値	機　能
18H	43H	DS：BX＝UCWのアドレス UCWのGBCPC＝パレット・レジスタにセットするカラー・コード(図4-5参照)	なし	「パレット・レジスタの設定」 パレット・レジスタにカラー・コードの設定を行う．モノクロ・モード時は表示画面の選択，合成になる．
	44H	DS：BX＝UCWのアドレス UCWのGBBCC＝セットするボーダ・カラー・コード D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 GBBCC 0 G R B 0 0 0 0		「ボーダ・カラーの設定」 標準ディスプレイを使用している場合はボーダ・カラーの設定ができる．専用高解像度ディスプレイでは，画面上下の色が出ない．
	45H	CH＝対象とする描画画面の指定(図4-6参照) ES＝描画パターン・バッファのセグメント・アドレス DS：BX＝UCWのアドレス GBON＿PTN＝3画面同時書き込み時の描画画面ナンバと描画オペレーション・モードの指定 GBDOTU＝単一画面の描画オペレーション・モードの指定 $GBSX_1$＝描画開始アドレスX座標 $GBSY_1$＝描画開始アドレスY座標 $GBLNG_1$＝書き込み長さ GBWDPA＝描画パターン・バッファの開始アドレス	なし	「ドットの書き込み」 グラフィック画面にドットを描く，描画画面，描画バンクを指定できる．
	46H	CH＝対象とする描画画面の指定 ES＝読み出しバッファ(1～3)のセグメント・アドレス DS：BX＝UCWのアドレス 使用できるUCW $GBSX_1$＝描画開始アドレスX座標 $GBSY_1$＝描画開始アドレスY座標 $GBLNG_1$＝書き込み長さ $GBRBUF_1$＝読み出しバッファ1の開始アドレス $GBRBUF_2$＝読み出しバッファ2の開始アドレス $GBRBUF_3$＝読み出しバッファ3の開始アドレス	なし D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 　　　　0 0 0 0 D_5, D_4＝描画画面ナンバ "00"　P1 "01"　P2 "10"　P3 "11"　P1/P2/P3 D_6＝描画範囲 "0"：LOWER/ALL "1"：UPPER D_7＝解像度モード "0"：標準解像度 "1"：専用高解像度	「ドットの読み出し」 指定された描画画面から，読み出しバッファにドット単位の読み出しを行う．
	47H	CH＝対象とする描画画面の指定(図4-6参照) DS：BX＝UCWのアドレス GBON＿PTN＝3画面同時書き込み時の描画画面ナンバと描画オペレーション・モードの指定 GBDOTU＝単一画面オペレーション・モードの指定 GBDSP＝描画開始方向 $GBSX_1$＝描画開始アドレスX座標 $GBSY_1$＝描画開始アドレスY座標 $GBSX_2$＝描画終了アドレスX座標 $GBSY_2$＝描画終了アドレスY座標 GBLPTN＝線種パターン GBDTYP＝描画タイプ	なし	「直線，矩形の描画」 指定された描画画面に直線，矩形を書き込む．描画に使用される線種パターンも指定できる．

INT	AH	入力パラメータ	戻り値	機　　能
18H	48H	CH＝対象とする描画画面の指定 （「直線，矩形の描画」と同じ） DS：BX＝UCW のアドレス GBON＿PTN＝3 画面同時書き込み時の描画画面ナンバと描画オペレーション・モードの指定 GBDOTU＝単一画面の描画オペレーション・モードの指定 GBDSP＝描画開始方向 $GBSX_1$＝描画開始アドレス X 座標 $GBSY_1$＝描画開始アドレス Y 座標 GBLNG＝描画総ドット数 GBMDOT＝マスキング・ドット数 GBCIR＝半径 GBLPTN＝線種パターン GBDTYP＝描画タイプ	なし	「円弧の描画」 指定された描画画面に円弧を書き込む．描画に使用される線種パターンも指定できる．
	49H	CH＝対象とする描画画面の指定 （「直線，矩形の描画」と同じ） DS：BX＝UCW のアドレス GBON＿PTN＝3 画面同時書き込み時の描画画面ナンバと描画オペレーション・モードの指定 GBDOTU＝単一画面の描画オペレーション・モードの指定 GBDSP＝描画開始方向 $GBSX_1$＝描画開始アドレス X 座標 $GBSY_1$＝描画開始アドレス Y 座標 $GBLNG_1$＝第 1 描画方向ドット数 $GBLNG_2$＝第 2 描画方向ドット数 GBMDOTI＝基本パターン情報	なし	「グラフィック文字の描画」 指定された描画画面にグラフィック文字を書き込む．
	4AH	CH＝描画タイミング・モードの設定 06H＝フラッシュ描画 16H＝フラッシュレス描画	なし	「描画モードの設定」 描画画面に対する GDC からの書き込みタイミングの設定． フラッシュ描画＝表示期間と，VRAM リフレッシュ期間を除いた時間に，GDC が VRAM をアクセスできる． フラッシュレス描画＝表示期間でも GDC が VRAM をアクセスする．フラッシュ描画にくらべて約 5 倍の書き込み速度が得られるが，画面にノイズが出る．

●使用上の注意

使用の前に，スタック・エリアをを 30 バイト確保し，SS，SP をセットする．
CPU のステータス・フラグのうち，IF（割り込み許可）＝セット，TF（シングル・ステップ・モード）＝クリアの状態にしておく．
描画データ等の受け渡し，保存のために約 80 バイトの制御領域（UCW）を確保しなければならない．UCW はユーザの好きなアドレス（セグメント/オフセット）を指定できる．UCW の内容は以下図 4-3 のとおり．

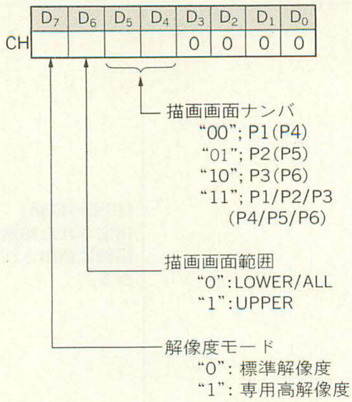
〈図 4-4〉表示領域の設定



〈図 4-5〉パレット・レジスタにセットするカラー・コード

$D_7 D_6 D_5 D_4$	$D_3 D_2 D_1 D_0$	$D_7 D_6 D_5 D_4$	$D_3 D_2 D_1 D_0$	$D_7 D_6 D_5 D_4$	$D_3 D_2 D_1 D_0$	$D_7 D_6 D_5 D_4$	$D_3 D_2 D_1 D_0$
#6	#7	#4	#5	#2	#3	#0	#1
O G R B	O G R B	O G R B	O G R B	O G R B	O G R B	O G R B	O G R B
←1 バイト→ GBCPC		←1 バイト→		←1 バイト→		←1 バイト→	

〈図 4-6〉 ドットの描き込み



D_7	D_6	D_5	D_4	描画画面と大きさ	
0	0	0	0	P1(0〜16K)	16KB
		0	1	P2(32K〜48K)	16KB
		1	0	P3(64K〜80K)	16KB
		1	1	P1/P2/P3	48KB
0	1	0	0	P4(16K〜32K)	16KB
		0	1	P5(48K〜64K)	16KB
		1	0	P6(80K〜96K)	16KB
		1	1	P4/P5/P6	48KB
1	0	0	0	P1(0〜32K)	32KB
		0	1	P2(32K〜64K)	32KB
		1	0	P3(64K〜96K)	32KB
		1	1	P1/P2/P3	96KB

RS-232-C BIOS

本体に内蔵されている RS-232-C インターフェースのための BIOS で，拡張 RS-232-C インターフェースは拡張ボード上の ROM によってサポートされます．

調歩同期式のみの 75〜9600bps までサポートされています．ブレーク信号の送受信はサポートされていません．

受信データは割り込み制御でバッファリングされますが，送信データはポーリングにより送出されます．受信は CTRL-S/CTRL-Q による X フロー制御が可能ですが，送信はいっさいフロー制御は行われません(図 4-7，p.128)．

割り込み番号 INT	機能番号 AH	入力パラメータ	戻り値	機　能
19H	00H	AL＝トランスファ・レート (図 4-8 参照) BH＝送信タイム・アウト時間 (01H〜FFH・単位は 500ms) BL＝受信タイム・アウト時間 (01H〜FFH・単位は 500ms) CH＝μPD8251 のモード設定コマンド CL＝μPD8251 のコマンド指定 ES：DI＝受信バッファのアドレス DX＝受信バッファ・サイズ	AH＝リターン・コード 00H＝正常終了	「RS-232-C BIOS の初期化」 RS-232-C インターフェース，BIOS の初期化．
	01H	AL＝トランスファ・レート (RS-232-C BIOS の初期化のトランスファ・レートを参照) BH＝送信タイム・アウト時間 (01H〜FFH・単位は 500ms) BL＝受信タイム・アウト時間 (01H〜FFH・単位は 500ms) CH＝μPD8251 のモード設定コマンド CL＝μPD8251 のコマンド指定 ES：DI＝受信バッファのアドレス DX＝受信バッファ・サイズ	AH＝リターン・コード 00H＝正常終了	「フロー制御を伴う初期化」 X パラメータでのフロー操作に対応した，RS-232-C インターフェース，BIOS の初期化．
	02H	なし	AH＝リターン・コード 00H＝正常終了 01H＝RS-232-C の初期化が行われていない 02H＝受信バッファがオーバ・フローした CX＝受信データ長	「受信データ長の取得」 受信バッファ内の有効なデータ長を得る．

INT	AH	入力パラメータ	戻り値	機　　能
19H	03H	AL＝送信データ	AH＝リターン・コード 00H＝正常終了 01H＝RS-232-C の初期化が行われていない 02H＝受信バッファがオーバ・フローした 03H＝送受信処理において，μPD8251 からの送受信可のステータスを引き取れなかった	「データの送信」 データを送信する．
	04H	なし	AH＝リターン・コード 00H＝正常終了 01H＝RS-232-C の初期化が行われていない 02H＝受信バッファがオーバ・フローした 03H＝送受信処理において，μPD8251 からの送受信可のステータスを引き取れなかった CH＝受信データ CL＝データ受信時のステータス	「データの受信」 受信され，バッファにためられたデータを読み出す．
	05H	CL＝μPD8251 へ出力するコマンド情報	AH＝リターン・コード 00H＝正常終了 01H＝RS-232-C の初期化が行われていない 02H＝受信バッファがオーバ・フローした	「μPD8251 へのコマンド出力」 μPD8251 へ指示されたコマンドを出力する．
	06H	なし	AH＝リターン・コード 00H＝正常終了 01H＝RS-232-C の初期化が行われていない 02H＝受信バッファがオーバ・フローした CH＝μPD8251 ステータス情報 CL＝モデム・ステータス (図 4-9 参照)	「ステータスの取得」 μPD8251 ステータスと，モデム・ステータスを得る

ノーマル・モード

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
DSR	BD	FE	OE	PE	TXE	$\overline{CS}$	$\overline{CD}$

μPD 8251 ステータス・バイト（D_7〜D_2）／システム・ポート・ステータス（D_1〜D_0）

ハイレゾ・モード

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
DSR	BD	FE	OE	PE	CI	$\overline{CS}$	$\overline{CD}$

μPD 8251 ステータス・バイト（D_7〜D_3）／モデム・ステータス（D_2〜D_0）

〈図 4-8〉転送レートの指定

トランスファ・レート(AL)	00H	01H	02H	03H	04H	05H	06H	07H
伝送速度(ボーレート，bps)	75	150	300	600	1200	2400	4800	9600

注：AL に 08H 以上の値を設定した場合は 1200bps とみなされる

〈図 4-9〉ステータスの取得

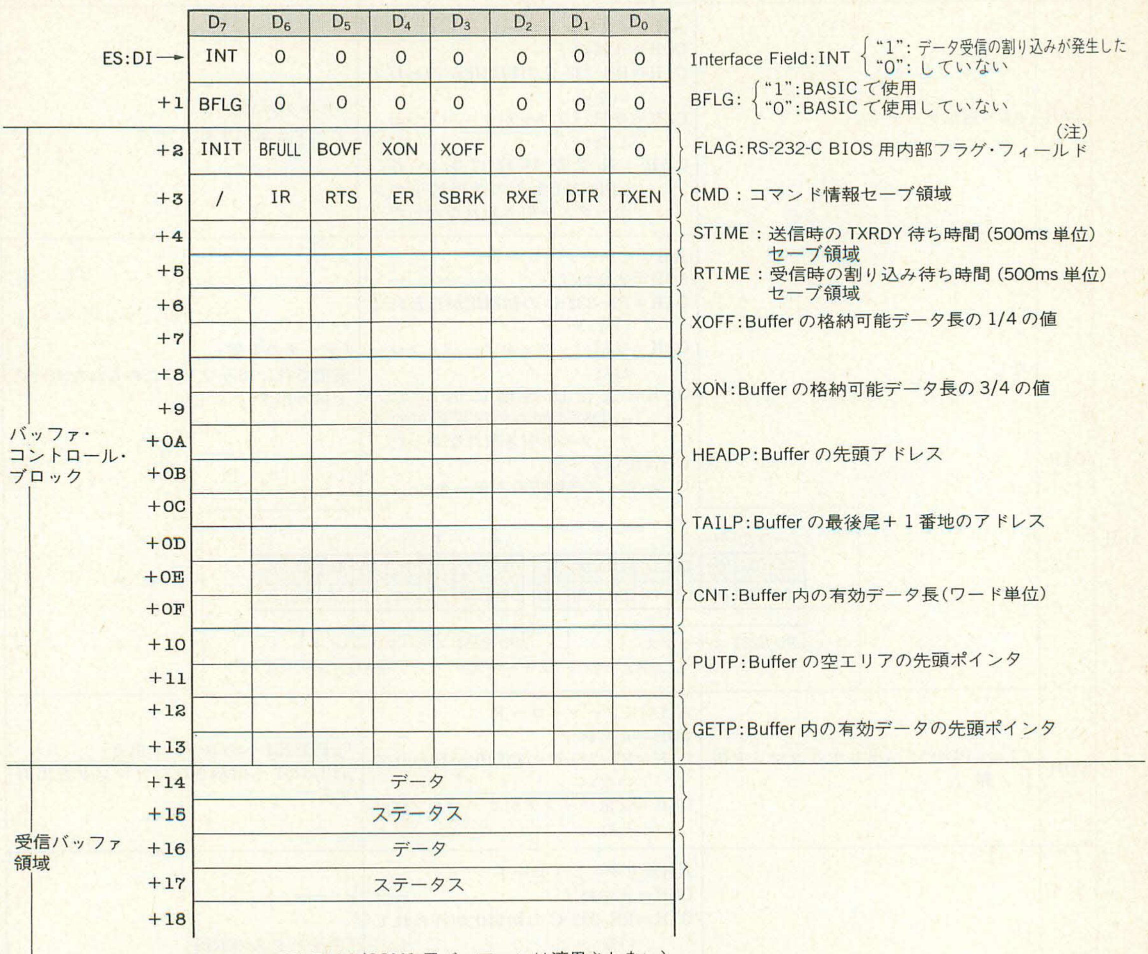
ビット	略　称	1	0
D_7	DSR	Data Set Ready ON	Data Set Ready OFF
D_6	BD	ブレーク状態検出あり	ブレーク状態検出なし
D_5	FE	フレーミング・エラー発生	フレーミング・エラーなし
D_4	OE	オーバーラン・エラー発生	オーバーラン・エラーなし
D_3	PE	パリティ・エラー発生	パリティ・エラーなし
D_2	TXE	送信バッファ・エンプティ	送信バッファ・フル
D_1	RXRDY	受信レディ(受信キャラクタあり)	受信ビジー
D_0	TXRDY	送信レディ	送信ビジー

CL＝モデム・ステータス

ビット	略称	1	0
D_7	$\overline{CI}$	着呼なし	着呼あり
D_6	$\overline{CS}$	送信不可	送信可
D_5	$\overline{CD}$	受信キャリア検出なし	受信キャリア検出
D_4	—		
D_3	—		
D_2	—		
D_1	—		
D_0	—		

注：PC9801 では，CI はサポートされていない

〈図4-7〉 RS-232-C BIOS



*:BFLG:BASIC FLAG(COM1用バッファには適用されない)
FLAG:RS-232-C BIOS用内部フラグ・フィールドの内容

INIT:RS-232-Cインタースェース(μPD8251)の初期化すみ
BFULL:受信バッファがFULL
BOVF:受信バッファ・オーバ・フローが発生
XON:XON処理(CTRL-S出力)を行う(3/4)
XOFF:XOFF処理(CTRL-Q出力)を行う(1/4)

プリンタBIOS

ノーマル・モードのプリンタ・インターフェースは，セントロニクス準拠で，フル・セントロニクス規格の簡易版になっています．印字データ出力は，I/Oが終了するまでBIOS内部でループします．

割り込み番号 INT	機能番号 AH	入力パラメータ	戻り値	機能
1AH	10H	なし	AH＝ステータス情報 00H＝プリンタBUSY 01H＝データ送信可	「プリンタBIOSの初期化」 プリンタ・インターフェース，ステータス情報の初期化

INT	AH	入力パラメータ	戻り値	機　　能
1AH	11H	AL＝出力する1バイト・データ	AH＝ステータス情報 00H＝プリンタ BUSY 01H＝データ送信可 02H＝タイム・アウト，データ未出力	「データの出力」 プリンタ・インターフェースが送信可能になるまで待って，データを出力する．
	12H	なし	AH＝ステータス情報 00H＝プリンタ BUSY 01H＝データ送信可	「ステータスの取得」 現在のプリンタのステータス情報を取得する．
	30H	CX＝データ長 ES：BX＝データ・バッファのアドレス	AH＝ステータス情報 00H＝プリンタ BUSY 02H＝タイム・アウト，データ未出力	「データの出力(複数バイト)」 指定したバッファ上の，指定した長さのデータをプリンタに出力する．

DISK BIOS

DISK BIOSは，フロッピ・ディスクやハード・ディスクの区別なくINT 1BHで呼び出します．AHレジスタの機能コードで読み書き等の機能を指定し，ALレジスタにドライブ種別・ユニット番号を入れて，フロッピ・ディスク／ハード・ディスクの区別を行います．

DISK BIOS機能一覧を図4-10に示します．

◈ 共通入力

DISK BIOS呼び出しのときのパラメータは共通部分が多くなっていますので，まとめて解説します．

● **AH＝BIOSコマンド識別コード**

コマンド(機能)はAHレジスタの下位ビット(0〜3)で指定します．

上位ビット(4〜7)はコマンド実行時の動作の指定をしますが，コマンドやデバイスにより受け付けられるパラメータが若干違います．

〈図4-10〉DISK BIOSの機能

AHレジスタ	機　　能	1MBFD	640KBFD	1M, 640MB 両用FD, RAMドライブ	320KBFD	固定ディスク	SCSI
01H	ベリファイ	○	○	←	○	○	○
02H	診断のための読み出し	○	×	←	×	×	○
03H	初期化	○	○	←	○	○	○
04H	センス	○	○	○	○	○	○
05H	データの書き込み	○	○	←	○	○	○
06H	データの読み出し	○	○	←	○	○	○
07H	シリンダ0へのシーク	○	○	←	×	○	○
09H	デリーテッド・データの書き込み	○	×	←	×	×	○
0AH	IDの読み出し	○	○	←	×	×	○
0CH	デリーテッド・データの読み出し	○	×	←	×	×	○
0DH	トラックのフォーマット	○	○	←	○	○	○
0EH	動作モードの設定	×	×	○	○	×	×
0FH	リトラクト	×	×	×	×	○	○
10H	シーク	○	○	←	×	×	×
83H	モータ停止モードの設定	×	×	○	×	×	×
83H	初期化	×	×	○	×	×	×

注：←はアクセス・モードに応じて1MBFDの機能や640MBFDの機能が使えることを示す

〈図4-11〉BIOSコマンド識別コード(FDD)

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
MT	MF	$\bar{r}$	SEEK	×	×	×	×

D_7(MT)　：シングル・トラック"0"/ マルチトラック"1"* の指定
　　(*同一シリンダの両面トラックのみ)
D_6(MF)　：単密度(FMモード)"0"/ 倍密度(MFMモード)"1"の読み出し指定
$D_5(\bar{r})$　：8回リトライする"0"/ リトライなし"1"の指定
D_4(SEEK)：シーク動作を行う"1"/ 現在のトラック位置"0"からの読み出し指定

コマンドに該当しないコードが指定された場合は正常終了(CF＝"0")にする．

〈図4-12〉BIOSコマンド識別コード(HDD)

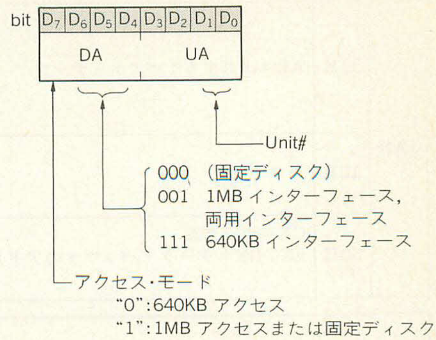
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
n	e	r	×	×	×	×	×

r＝リトライ・フラグ(SASIのみ)
e＝エラー通知ビット(SCSIのみ)
n＝コマンドにより指定するものもある

〈図 4-13〉アクセス・デバイスの種類の設定

デバイス	アクセス・デバイス	ユニット番号
0×H	SASI ドライブ(相対セクタ・アドレス)	0~6
1×H	1MB/640KB 両用ドライブでの 640KB モード	0~3
2×H	SCSI ドライブ(相対セクタ・アドレス)	0~6
5×H	320KB 専用ドライブ	0~3
7×H	640KB 専用ドライブ	0~3
8×H	SASI ドライブ(絶対セクタ・アドレス)	0~1
9×H	1MB(専用/両用を含む)ドライブ	0~3
A×H	SCSI ドライブ(絶対セクタ・アドレス)	0~6
F×H	640KB 専用ドライブ	0~3

〈図 4-14〉アクセス・デバイス識別コード



▶フロッピ・ディスク(1MB/640KB)

BIOS コマンド識別コード・FDD を図 4-11 に示しました.

▶ハード・ディスク

BIOS コマンド識別コード・HDD を図 4-12 に示しました.

● AL=デバイス・タイプ識別コード

下位ビット(0~3)でユニット(ドライブ)番号を指定(UA)します.

上位ビット(4~7)でデバイスの種類指定(DA)します.

アクセス・デバイスの種類の指定を図 4-13 に,アクセス・デバイス識別コードを図 4-14 に示します.

● BX=データ長

フロッピ・ディスクの場合は,読み書きするデータの長さ(バイト)を指定します.

ハード・ディスクの場合は,256(512)バイト単位で指定します.

● CH=セクタ長

セクタ長は下記のように 0~3 の数字で表されます.ハード・ディスクや 320KB ドライブの場合は無効です.

CH バイト数
 0 128 バイト
 1 256 バイト
 2 512 バイト
 3 1024 バイト

● CL(CX)=シリンダ番号

シリンダ番号で,デバイスによって下記のように指定範囲が違います.ハード・ディスクの場合は CX レジスタ(16 ビット)を使用します.

320KB 片面 0~34/両面 0~39
640KB 0~79
1MB 0~76
HDD 使用するドライブによって変わる

● DH=ヘッド番号

ヘッド番号を次のとおりに指定します.

FDD 0~1
HDD 使用するドライブによって変わる

● DL=セクタ番号

セクタ番号を指定します.

320KB 1~16
640KB 1~16
1MB 1~26
SASI 0~32(PC9801-27)
SCSI 使用するドライブによって変わる

● ES:BP=データ・バッファ領域先頭アドレス

読み書きするデータの先頭アドレスを指定します.

◈ 共通出力

▶ CF=終了条件
 0=正常終了
 1=異常終了

▶ AH=ステータス情報

◈ フロッピ・ディスク・ステータス情報一覧

ステータス情報一覧を図 4-15 に示します.

◈ ハード・ディスク・ステータス情報一覧

コマンド実行終了に得られるステータス情報は図 4-16 に示すような意味があります.

◈ 1MB/640KB 両用インターフェース

1MB/640KB 両用タイプのインターフェースは,1MB/640KB 自動切り替えインターフェース(1MB インターフェース・モード)と,640KB 専用インターフェース(640KB インターフェース・モード)の二つの動作モードがあります.

640KB インターフェース・モードは,640KB 専用インターフェースと同等の機能を持ち,DMA や割り込み等も同じになります.1MB インターフェース・モードでは,1MB 専用インターフェースの機能を含んだうえに,640KB のフロッピ・ディスクを扱う機能も持っています.

〈図 4-15〉 ステータス情報(FDD)

CF	AH		説明		FDC STATUS との対応		詳細
	16進	ビット	略称	内容	$ST_{1\sim3}$	$D_0\sim D_6$	
0	00H	0000××××	NT	Normal end			
0	00H	0000××××	RY	Ready(センス)	ST_3	D_5	
0	10H	0001××××	CM	Control Mark (1MBFD)	ST_2	D_6	DDAM(Deleted Data Addres Mark)を検出した*1
0	10H	0001××××	WP	Write Protect (センス)	ST_3	D_6	媒体はセットされているが，ライトプロテクト状態
1	20H	0010××××	DB	DMA Boundary			メモリ・アドレスがバンクにまたがるか，奇数番地から始まるように指定した
1	30H	0011××××	EN	ENd of cylinder	ST_1	D_7	1回の動作の転送容量を越えてデータ長(DTL)を指定した
1	40H	0100××××	EC	Equipment Check	ST_0	D_7	デバイスからFault信号を受けとった*2,*3,*4
1	50H	0101××××	OR	OverRun	ST_1	D_4	セクタ,メモリ間のデータ転送時に,一定時間内にデータ転送が終了できなかった*5
1	60H	0110××××	NR	Not Ready	ST_0	D_3	ユニットがノット・レディ状態
1	70H	0111××××	NW	Not Writable	ST_1	D_1	コマンド実行開始時,Write Protected信号がオン
1	80H	1000××××	ER	ERror			
1	90H	1001××××	TO	Time Out			
1	A0H	1010××××	DE	DataError(ID)	ST_1 ST_2	D_5 D_5	ID読み出し時にCRCエラーが発生した
1	B0H	1011××××	DD	DataError(Data)	ST_1 ST_2	D_5 D_5	
1	C0H	1100××××	ND	No Data	ST_1	D_2	トラック内に，指定されたセクタが見つからなかった
1	D0H	1101××××	BC	Bad Sylinder	ST_2	D_1	
1	E0H	1110××××	MA	Missind Address mark(ID)	ST_1 ST_2	D_0 D_0	トラック内に，指定されたセクタが見つからず，かつIDが1個もなかった，もしくは指定されたセクタのID検出後データ部のアドレス・マークを一定時間内(1ms/8MHz)に検出できなかった
1	F0H	1111××××	MD	Missind address mark(Data)	ST_1 ST_2	D_0 D_0	
0	01H	××××0001		両面媒体がセットされている	ST_3	D_3	
1	08H	00001×××	CD	Corrected Data			
1	78H	00111×××	IA	Illegal disk Address			
1	88H	10001×××		Direct access an alternate track			
1	B8H	10111×××		Data Error			
1	C8H	11001×××		Seek error			
1	D8H	11011×××		代替トラックが読めない			

＊1 「データの読み出し　AH＝06H」ではそのセクタを読み出し後正常終了する.
＊2 「データの書き込み　AH＝05H」の場合は，データは書き込まれる.
＊3 「トラックのフォーマット　AH＝0DH」の場合は書き込み終了時にチェックされる.
＊4 「シリンダ0へのシーク　AH＝07H」の場合は一定時間内にトラック0を確認されなかった場合.
＊5 「データの書き込み　AH＝05H」の場合はそのセクタを書き込み後，ORとなる.

ステータスのビット7~5は，CF＝0のとき“000”，CF＝1のとき“000”以外となる.
ビット3~0は，センス・コマンドで装置種別が通知される(CF＝0のとき有効).

ビット0 { “0”：片面媒体がセットされている
　　　　 “1”：両面媒体がセットされている(1MB/640KB両用ドライブでは常に両面)

ビット2 { “0”：40シリンダ・モード，もしくは1MFDアクセス・モード
　　　　 “1”：80シリンダ・モード

〈図 4-16〉ステータス情報(HDD)

エラー・ステータス情報			リトライ処理
略称	内容	AHの内容	
NT	Normal end	0 0 0 0 × × × ×	
DB	DMA Boundary	0 0 1 0 × × × ×	コマンド使用上に誤りがある
EN	End of cylinder	0 0 1 1 × × × ×	
EC	Equipment Check	0 1 0 0 × × × ×	再試行する
OR	Over Run	0 1 0 1 × × × ×	
NR	Not Ready	0 1 1 0 × × × ×	
NW	Not Writable	0 1 1 1 × × × ×	
TO	Time Out	1 0 0 1 × × × ×	
DE	Data Error(ID)	1 0 1 0 × × × ×	Recalibrate → Seek →リード/ライト系コマンド
DD	Data Error(Data)	1 0 1 1 × × × ×	
ND	No Data	1 1 0 0 × × × ×	
MA	Missing Address mark(ID)	1 1 1 0 × × × ×	
MD	Missing Address mark(Data)	1 1 1 1 × × × ×	
IA	Illegal Disk Error	1 0 0 1 1 × × ×	
	Defect List Error	1 0 0 1 1 × × ×	
	No Defect Spare Location	1 0 1 0 1 × × ×	
	Seek Error	1 1 0 0 1 × × ×	

640KBインターフェース・モードにするためには，ディップ・スイッチ(SW_{3-1}/SW_{3-2})で指定して，再立ち上げする必要があります．通常は1MBインターフェース・モードで使われます．この場合は，DMAや割り込み等も1MB専用インターフェースと同じものが使用されます．

1MBインター・フェース・モードで640KBのフロッピ・ディスクをアクセスするときは，デバイス・タイプに70H～73Hではなく10H～13Hを使用します．

◈ 320KBインターフェース

320KBフロッピ・インターフェースは，PC9801/E/F/Mでのみ使用可能です．デバイス・タイプ(DA)には50H～53Hを使用します．

1MB/640KB両用ドライブでは，320KBのディスクを読むモードがあります．これは，320KB(2D)と640KB(2DD)のメディアの記憶密度，記憶方式が同じでトラックの密度だけが違うため，トラックの移動ステップを2倍にすることで，640KBモードで2Dを読み出すしくみです．しかし，ヘッドの幅が2Dと2DDでは違うために，書き込みは保証されません．

◈ 固定ディスクBIOSコマンド

● エラー・リトライ処理

エラー・リトライは，ハードウェアで8回まで行える設定にできます．

● 相対アドレス

セクタの指定方法には，シリンダ・ヘッド・セクタを別々に指定する「絶対セクタ・アドレス」と，0～2097152(21ビット)のシーケンシャル番号で指定する「相対アドレス」の2種類でアクセスできます．

絶対セクタ・アドレスでは，CX(シリンダ)，DH(ヘッド)，DL(セクタ)で指定し，相対セクタ・アドレスでは，DH：DL：CH：CL(SASIの場合使用するのは21ビット)で指定します．

● IDE・HDDの違い

IDEではリトラクトはサポートしていません．コマンド実行時は何もせずに，またエラーも出しません．トラック・フォーマット・コマンドを実行しても物理フォーマットはされず，0シリンダ，0トラックの先頭16KバイトにE5Hを書き込むだけです．

物理セクタは512バイトに固定されているので，セクタ長を256バイトに指定した場合はソフトウェア・エミュレートされます．

98NOTE等では，一定時間アクセスがないときに，モータOFFにできます．次にアクセスした場合は自動的にモータをONにしますが，回転が安定するまで通常4～5秒かかるために，最初のアクセスは遅くなります．

IDEのHDDを持った機種を判定するには，図4-17に示すシステム共通域のビットを確認します．

◈ RAMドライブBIOS

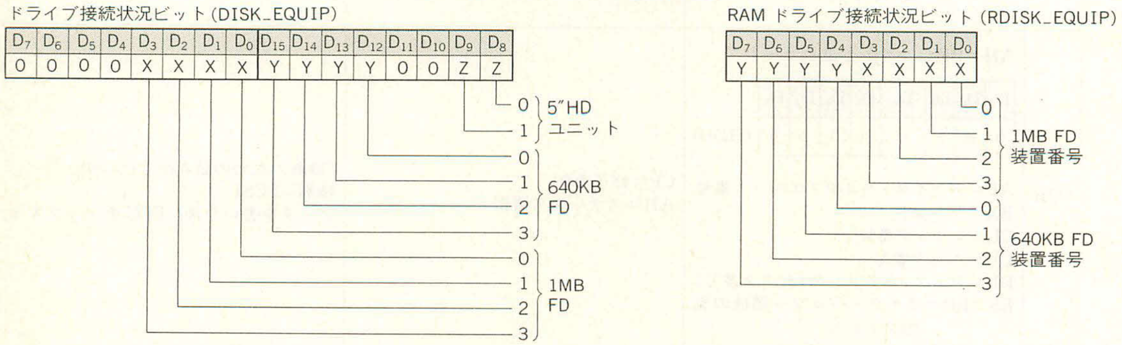
RAMドライブは，1MB/640KB両用タイプのフロッピ・ディスク・ドライブとBIOSレベルで互換性があるために，ソフトウェアからは同等に扱うことができます．実際に利用する場合は，1MB/640KB両用タイプ用のBIOSを使用します．

〈図 4-17〉システム共通域のビットの確認

		000：480H	0000：480H ビット 7
NS	固定ディスク内蔵モデル	≠00	＝1
NS/E	FD モデル	≠00	＝0
NC	FD モデル＋従来固定ディスク	≠00	＝0
他機種	固定ディスク内蔵モデル	＝00	＝1
	FD モデル	＝00	＝0
	FD モデル＋従来固定ディスク	＝00	＝0

システム共通域　0000：480H　ビット 7＝1　かつ，0000：457H≠00

〈図 4-18〉RAM ドライブ装置の確認



● RAM ドライブ装置の確認方法

RAM ドライブ装置の確認は，システム共通領域(図 4-18)のドライブ認識ビット(DISK_EQUIP)，RAM ドライブ接続状況認識ビット(RDISK_EQUIP)を参照します．

DISK_EQUIP は，実際の FDD だけでなく，RAM ドライブの搭載でもビットが立つので，RDISK_EQUIP のビットと照らし合わせて，ともに同じ装置番号にビットが立っていれば，RAM ドライブになります．

以上の情報は，装置の有無を示すもので，RAM ドライブのフォーマット状態は，BIOS を使用して調べる必要があります．

ハード・ディスク BIOS

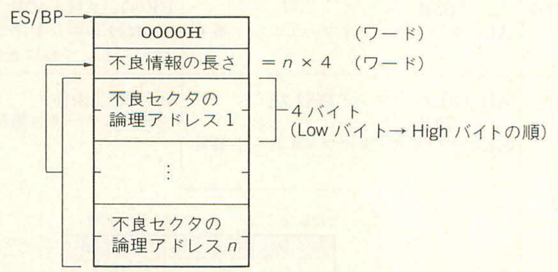
割り込み番号 INT	機能番号 AH	入力パラメータ	戻り値	機能
1BH	01H	AH＝BIOS コマンド識別コード D7 D6 D5 D4 D3 D2 D1 D0 MT MF r Seek 0 0 0 1 (FDD) 0 e r 0 0 0 0 1 (HDD) AL＝デバイス・タイプ・ユニット番号 BX＝データ長 CL(CX)＝シリンダ番号 DH＝ヘッド番号 DL＝セクタ番号 CH＝セクタ長＊1 ES：BP＝データ・バッファ領域の先頭アドレス	CF＝終了条件 AH＝ステータス情報	「ベリファイ」 種別＝1MB/640KB/2MODE/320KB/SASI/SCSI 指定したデバイス・タイプ・ユニットの指定された場所のデータを読み出すが，データのメモリへの転送は行われない．

＊1　HDD 時は，シリンダ番号は「CX レジスタ」を使用し，セクタ長(CH レジスタ)は使用しない
＊2　HDD 時のセンス情報
＊3　ES：BP＝フォーマットするセクタの「シリンダ番号」，「ヘッド番号」，「セクタ番号」，「セクタ長」を1トラック当たりのセクタ数分指定する．BX＝ES：BP で指定したデータ列の長さ

<table>
<tr><th>INT</th><th>AH</th><th>入力パラメータ</th><th>戻り値</th><th>機　　能</th></tr>
<tr><td rowspan="6">1BH</td><td>02H</td><td>AH＝BIOS コマンド識別コード
<table>
<tr><td>D_7</td><td>D_6</td><td>D_5</td><td>D_4</td><td>D_3</td><td>D_2</td><td>D_1</td><td>D_0</td><td></td></tr>
<tr><td>0</td><td>MF</td><td>r</td><td>Seek</td><td>0</td><td>0</td><td>1</td><td>0</td><td>(FDD)</td></tr>
</table>
AL＝デバイス・タイプ・ユニット番号
BX＝データ長
CL＝シリンダ番号
DH＝ヘッド番号
DL＝セクタ番号
CH＝セクタ長
ES：BP＝データ・バッファ領域の先頭アドレス</td><td>CF＝終了条件
AH＝ステータス情報</td><td>「診断のための読み出し(フロッピ・ディスク)」
種別＝1MB/2MODE
インデックス・マークの直後から読み取りを開始し，IDのエラー，データ部のエラーがあっても読み取りを続けることを除いて「データの読み出し」と同じ機能を持つ．</td></tr>
<tr><td>02H</td><td>AH＝BISO コマンド識別コード
<table>
<tr><td>D_7</td><td>D_6</td><td>D_5</td><td>D_4</td><td>D_3</td><td>D_2</td><td>D_1</td><td>D_0</td><td></td></tr>
<tr><td>0</td><td>e</td><td>r</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>(HDD)</td></tr>
</table>
AL＝デバイス・タイプ・ユニット番号
BX＝データ長
CL＝シリンダ番号
DH＝ヘッド番号
DL＝ディスク・アドレス(セクタ番号)
ES：BP＝データ・バッファ領域の先頭アドレス</td><td>CF＝終了条件
AH＝ステータス情報</td><td>「診断のための読み出し(SCSI)」
種別＝SCSI
データを書いた後，ECC チェックする．</td></tr>
<tr><td>03H</td><td>AH＝BIOS コマンド識別コード
<table>
<tr><td>D_7</td><td>D_6</td><td>D_5</td><td>D_4</td><td>D_3</td><td>D_2</td><td>D_1</td><td>D_0</td><td></td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>(FDD)</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>(HDD)</td></tr>
</table>
AL＝デバイス・タイプ・ユニット番号</td><td>CF＝終了条件
AH＝ステータス情報</td><td>「初期化」
種別＝1MB/640KB/2MODE/320KB/S ASI/SCSI
ディスク装置，コントローラの初期化を行う．</td></tr>
<tr><td>04H</td><td>AH＝BIOS コマンド識別コード
<table>
<tr><td>D_7</td><td>D_6</td><td>D_5</td><td>D_4</td><td>D_3</td><td>D_2</td><td>D_1</td><td>D_0</td><td></td></tr>
<tr><td>n</td><td>0</td><td>r</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>(FDD)</td></tr>
</table>
AL＝デバイス・タイプ・ユニット番号</td><td>CF＝終了条件
AH＝ステータス情報＊2</td><td>「センス(フロッピ・ディスク)」
種別＝1MB/640KB/2MODE/320KB
指定したデバイスの状態を調べる．</td></tr>
<tr><td>04H</td><td>AH＝BIOS コマンド識別コード
04H/44H/84H
AL＝デバイス・タイプ・ユニット番号</td><td>CF＝終了条件
AH＝ステータス情報
AH＝00H の場合(SASI)
<table>
<tr><td></td><td>D_7</td><td>D_6</td><td>D_5</td><td>D_4</td><td>D_3</td><td>D_2</td><td>D_1</td><td>D_0</td></tr>
<tr><td></td><td>d_7</td><td>d_6</td><td>d_5</td><td>d_4</td><td></td><td></td><td></td><td></td></tr>
<tr><td>接続されているデバイスの容量を通知する　5MB</td><td></td><td></td><td></td><td></td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>10MB</td><td></td><td></td><td></td><td></td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>20MB</td><td></td><td></td><td></td><td></td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>40MB</td><td></td><td></td><td></td><td></td><td>0</td><td>1</td><td>0</td><td>0</td></tr>
</table>
d_7 d_6 d_5 d_4 については「ステータス一覧」を参照
AH＝44H の場合(SCSI)
BX＝1(ソフト・セクタ)
＝2(ハード・セクタ)
AH＝84H の場合(SASI/SCSI)
BX＝セクタ長
CX＝シリンダ番号
DH＝ヘッド番号
DL＝セクタ数</td><td>「センス(ハード・ディスク)」
種別＝SASI/SCSI
デバイスの状態，諸元を読む．</td></tr>
</table>

INT	AH	入力パラメータ	戻り値	機　　能
1BH	05H	AH＝BIOS コマンド識別コード D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 MT MF r Seek 0 1 0 1 (FDD) 0 e r 0 0 1 0 1 (HDD) AL＝デバイス・タイプ・ユニット番号 BX＝データ長 CL(CX)＝シリンダ番号 DH＝ヘッド番号 DL＝セクタ番号 CH＝セクタ長＊1 ES：BP＝データ・バッファ領域の先頭アドレス	CF＝終了条件 AH＝ステータス情報	「データの書き込み」 種別＝1MB/640KB/2MODE/320K
	06H	AH＝BIOS コマンド識別コード D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 MT MF r Seek 0 1 1 0 (FDD) 0 e r 0 0 1 1 0 (HDD) AL＝デバイス・タイプ・ユニット番号 BX＝データ長 CL(CX)＝シリンダ番号 DH＝ヘッド番号 DL＝セクタ番号 CH＝セクタ長＊1 ES：BP＝データ・バッファ領域の先頭アドレス	CF＝終了条件 AH＝ステータス情報	「データの読み出し」 種別＝1MB/640KB/2MODE/320KB/SASI/SCSI 指定したデバイス・タイプ・ユニットから，指定された場所のデータを，指定されたアドレスへ読み出す．
	07H	AH＝BIOS コマンド識別コード D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 0 0 r 0 0 1 1 1 (FDD) 0 0 r 0 0 1 1 1 (HDD) AL＝デバイス・タイプ・ユニット番号	CF＝終了条件 AH＝ステータス情報	「シリンダ0へのシーク」 種別＝1MB/640KB/2MODE/SASI/SCSI 指定したデバイス・タイプ・ユニットに対して，0シリンダへシークさせる．
	09H	AH＝BIOS コマンド識別コード D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 MT MF r Seek 1 0 0 1 (FDD) AL＝デバイス・タイプ・ユニット番号 BX＝データ長 CL＝シリンダ番号 DH＝ヘッド番号 DL＝セクタ番号 CH＝セクタ長 ES：BP＝データ・バッファ領域の先頭アドレス	CF＝終了条件 AH＝ステータス情報	「デリーテッド・データの書き込み（フロッピ・ディスク)」 種別＝1MB/2MODE セクタのデータ・フィールドの Data Address Mark の代わりに Delated Data Address Mark を書き込むことを除いては「データの書き込み」と同じ機能を持つ．
	09H	AH＝BIOS コマンド識別コード 09H AL＝デバイス・タイプ・ユニット番号 BX＝データの長さ ES：BP＝DEFECT LIST の先頭アドレス (図 4-19 参照)	CF＝終了条件 AH＝ステータス情報	「不良セクタの代替(SCSI)」 不良セクタを代替する．

INT	AH	入力パラメータ	戻り値	機　能
1BH	0AH	AH＝BIOSコマンド識別コード D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 0 MF r Seek 1 0 1 0 (FDD) AL＝デバイス・タイプ・ユニット番号 CL＝シリンダ番号(AHのSEEKビットが1のときに意味を持つ) DH＝ヘッド番号	CF＝終了条件 AH＝ステータス情報 CH＝セクタ長 CL＝シリンダ番号 DH＝ヘッド番号 DL＝セクタ番号 機能＝「IDの読み出し(フロッピ・ディスク)」 種別＝1MB/640KB/2MODE 指定したデバイス・タイプ・ユニットの指定されたトラックで，最初に正常に読み出せるIDをID情報に格納する．	
	0AH	AH＝BIOSコマンド識別コード0AH AL＝デバイス・タイプ・ユニット番号 BH＝セクタ長 1＝256バイト 2＝512バイト 3＝1024バイト	CF＝終了条件 AH＝ステータス情報	「セクタ長指定(SCSI)」 セクタ長を指定する．
	0CH	AH＝BIOSコマンド識別コード D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 MT MF r Seek 1 1 0 0 (FDD) AL＝デバイス・タイプ・ユニット番号 BX＝データ長 CL＝シリンダ番号 DH＝ヘッド番号 DL＝セクタ番号 CH＝セクタ長 ES：BP＝データ・バッファ領域の先頭アドレス	CF＝終了条件 AH＝ステータス情報	「デリーテッド・データの読み出し(フロッピ・ディスク)」 種別＝1MB/2MODE セクタのデータ・フィールドの，「Data Address Mark」の代わりに「Deleted Data Address Mark」を扱うことを除いては「データの読み出し」と同じ機能を持つ．
	0CH	AH＝BIOSコマンド識別コード 0CH/8CH 0CH＝READ DEFECT DATA 8CH＝READ DEFECT NUMBER AL＝デバイス・タイプ・ユニット番号 BX＝DEFECT LISTの長さ (READ DEFECT DATA) ES：BP＝DEFECT LISTを格納する先頭アドレス (READ DEFECT DATA)	CF＝終了条件 AH＝ステータス情報 CX＝代替されている不良セクタの数 (READ DEFECT NUMBER)	「代替情報取得(SCSI)」 代替を行っているセクタの情報/数を返す．
	0DH	AH＝BIOSコマンド識別コード D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 0 MF r Seek −1 1 0 1 (FDD) AL＝デバイス・タイプ・ユニット番号 BX＝データ長＊3 CH＝セクタ番号 CL＝シリンダ番号 DH＝ヘッド番号 DL＝データ部への書き込みデータ・パターン ES：BP＝データ・バッファ領域の先頭アドレス＊3	CF＝終了条件 AH＝ステータス情報	「フォーマット(フロッピ・ディスク)」 種別＝1MB/640KB/2MODE/320KB 指定した1トラックを任意の形式でフォーマットする．

INT	AH	入力パラメータ	戻り値	機　　能
1BH	0DH	AH＝BIOS コマンド識別コード D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 d e r 0 1 1 0 1 (HDD) d＝0 トラック単位(SASI のみ) d＝1 ドライブ単位 D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 d x $\bar{r}$ x 1 1 0 1 "D_7"＝フォーマット単位の指定 0：トラック単位 1：デバイス単位 AL＝デバイス・タイプ・ユニット番号 BH＝インタリーブ・ファクタ(1〜16) CX＝シリンダ番号 DH＝ヘッド番号 DL＝0 ※ドライブ単位でフォーマットする場合は，CX＝0，DH＝0 とする．	CF＝終了条件 AH＝ステータス情報  〈図 4-19〉不良セクタの代替	「フォーマット(ハード・ディスク)」 種別＝SASI/SCSI 指定したトラック(SASI のみ)/ドライブをフォーマットする．
	0FH	AH＝BIOS コマンド識別コード D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 0 0 r 0 1 1 1 1 (HDD) AL＝デバイス・タイプ・ユニット番号	CF＝終了条件 AH＝ステータス情報	機能＝「リトラクト」 種別＝SASI/SCSI 未使用シリンダへ，ヘッドを退去させる．
	10H	AH＝BIOS コマンド識別コード D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 0 0 r 1 0 0 0 0 (FDD) AL＝デバイス・タイプ・ユニット番号 CL＝シリンダ番号	CF＝終了条件 AH＝ステータス情報	「シーク」 種別＝1MB/640KB/2MODE/320KB 指定したデバイス・タイプ・ユニットに対して，指定されたシリンダまでシークする．

1MB/640KB 両用
フロッピ・ディスク専用 BIOS

割り込み番号 INT	機能番号 AH	入力パラメータ	戻り値	機　　能
1BH	0EH	AH＝BIOS コマンド識別コード 0EH/8EH AL＝各ユニットの動作モード D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0 0 0 0 1 D_0＝Unit #0 D_1＝Unit #1 D_2＝Unit #2 D_3＝Unit #3 AH＝0E のとき，上記各 Unit の bit は "0"：片面モード，"1"：両面モード AH＝8E のとき，上記各 Unit の bit は "0"：48tpi モード，"1"：96tpi モード 注：各ビットの初期状態はすべて "1" である．これらの指定は，DA＝1×H のコマンドに対して有効となる．	CF＝終了条件 CF＝0，AH＝00H 正常終了 CF＝1，AH＝40H 異常終了	「動作モードの設定」 種別＝2MODE(1MB モード) 両用ドライブを 1MB モードで使用する際に，1MB/640KB のアクセス・モードを設定する．

INT	AH	入力パラメータ	戻り値	機　　能
1BH	83H	AH＝BIOS コマンド識別コード 83H AL＝デバイス・タイプ・ユニット番号 (PC9801/E/F/M では，7×H のときに正常終了となるので F×H を使用するほうが望ましい)	CF＝終了条件 CF＝0，AH＝00H 正常終了 CF＝1，AH＝40H 異常終了	「初期化」 種別＝2MODE(640KB モード) 両用ドライブを 640KB インターフェース・モード時に，AI を検出するように初期化する．
	83H	AH＝BIOS コマンド識別コード 83H AL＝デバイス・タイプ・ユニット番号	CF＝終了条件 CF＝0，AH＝00H 正常終了 CF＝1，AH＝40H 異常終了 (外付けユニットにこのコマンドを実行しても，つねに正常終了が返る)	「モータ停止モードの設定」 種別＝2MODE(1MB モード) 1MBFD のモータを自動的に ON/OFF するモードに設定する．
	84H	AH＝BIOS コマンド識別コード 84H AL＝デバイス・タイプ・ユニット番号	CF＝終了条件 AH＝ステータス情報	「センス」 種別＝2MODE 指定したデバイスの状態を調べる．

1MB インターフェース・モード時

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
					0	0	

・1MB フロッピ・ディスク BIOS「センス AH＝04H」と同じ

D_0＝"0"：片面媒体
　"1"：両面媒体 (1MB/640KB 両用ドライブでは常に両面)
D_3＝"0"：1MB ドライブ
　"1"：1MB/610KB 両用ドライブ

640KB インターフェース・モード時

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

・640KB フロッピ・ディスク BIOS「センス AH＝04H」と同じ

D_0＝"0"：片面モード
　"1"：両面モード
D_1＝"0"：AI あり (1MB ドライブのときは必ず 0)
　"1"：AI なし
D_2＝"0"：48tpi(40 シリンダ)モード
　"1"：96tpi(80 シリンダ)モード
　　640KB タイプのときのみ有効．
　　1MB タイプでは必ず 0 となる
D_3＝"0"：640KB ドライブ
　"1"：1MB/640KB 両用ドライブ
注：エラー発生時は，D_3～D_0 のうち両用タイプ・ビット (D_3) のみ有効

INT	AH	入力パラメータ	戻り値	機　　能
1BH	9EH	AH＝BIOS コマンド識別コード 9EH AL＝デバイス・タイプ・ユニット番号 (1CH～1DH/内蔵ドライブのみ)	CF＝終了条件 CF＝0，AH＝00H 正常終了 CF＝1，AH＝40H 異常終了	「2D モードに切り替える」
	9EH	AH＝BIOS コマンド識別コード 9EH AL＝デバイス・タイプ・ユニット番号 (1EH～1FH/内蔵ドライブのみ)	CF＝終了条件 CF＝0，AH＝00H 正常終了 CF＝1，AH＝40H 異常終了	「2DD モードに切り替える」

タイマ BIOS

タイマ BIOS は，カレンダ時計の設定・読み出し，インターバル・タイマの設定等を行います．

割り込み番号 INT	機能番号 AH	入力パラメータ	戻り値	機　　能
1CH	00H	ES：BX＝読み出した日付けを格納するデータ・バッファ(6 バイト)のアドレス	データ・バッファに格納される	日付け・時刻の読み出し 現在の年，月，曜日，日，時，分，秒をカレンダ時計 LSI から読み出して，データ・バッファに格納する(図 4-20 参照)． カレンダ時計 LSI への直接のアクセスは避けて，BIOS を使用すべきである．

INT	AH	入力パラメータ	戻り値	機　　能
1CH	01H	ES：BX＝読み出した日付け，時刻を格納されているデータ・バッファ(6バイト)のアドレス．	なし	「日付け・時刻の設定」 データ・バッファに格納されている，年，月，曜日，日，時，分，秒をカレンダ時計LSIへ設定する． データ・バッファ形式は「日付け・時刻の読み出しと同じ」
1CH	02H	CX＝インターバル・タイマ値 n (単位は10ms) $1 \leqq n \leqq 65536$ (n=65536はn=0と設定する) ES：BX＝ユーザのタイマ割り込み処理ルーチンのアドレス．	なし	「インターバル・タイマの設定(シングル・イベント) インターバル・タイマ値を設定し，起動する．設定値まで時間が経過すると割り込みを発生し，ユーザのタイマ割り込み処理ルーチンに制御を移す．1回の設定で1回だけインターバル・タイマが起動される(リスト4-1参照)．

〈図4-20〉日付け・時刻の読み出し

最下位番地←　　　　　　　　　　　　　　　→最上位番地

「10年」位	「1年」位	月	曜	「10日」位	「1日」位	「10時」位	「1時」位	「10分」位	「1分」位	「10秒」位	「1秒」位

1バイト←――ES:BX

項目	データ形式	範囲	バイト数
年	BCD	00H～99H	1
月	16進数	01H～0CH	1
曜	16進数	00H～06H	
日	BCD	01H～31H	1
時	BCD	00H～23H	1
分	BCD	00H～59H	1
秒	BCD	00H～59H	1

注：PC9801/E/F1, 2, 3/M2, 3/U2/VF0, 2, 4/UV2/PC-98XAでは，「年」は不揮発メモリに格納されるために，自動的に更新されない．また，うるう年もサポートされない．
「月」は月の大小を自動判別する

〈リスト4-1〉インターバル・タイマの設定(サンプル・プログラム)

```
/*
**  インターバルタイマーのテスト
**
**  ５秒間の間に何回メモリーアクセスができるか数える。
*/

#include <dos.h>
#include <stdio.h>

    int     endflg  = 0;

/*
**  インターバルタイマの設定
**
*/
void settimer(unsigned int count,void interrupt (*intprg)())
{
    struct REGPACK  reg;

    reg.r_ax = 0x0200;              /*  インターバルタイマの設定          */
    reg.r_cx = count;               /*  インターバルタイマー値 nx10ms     */
    reg.r_es = FP_SEG(intprg);      /*  タイマ処理ルーチンのセグメント    */
    reg.r_bx = FP_OFF(intprg);      /*  タイマ処理ルーチンのオフセット    */
    intr(0x1c,&reg);                /*  タイマＢＩＯＳをコール            */
}

/*
**  タイマー割り込み処理部
**
**  終了のフラグをたてる。
*/
void interrupt timer(void)
{
    endflg = 1;
}

void main(void)
{
    unsigned int    i,j,dmy=0;

    printf("簡易ベンチマーク （５秒お待ち下さい）＞");
    settimer(500,timer);            /*  インターバルタイマの設定 500x10ms   */
    for (i = 0; i < 60000; i++){
        for (j = 0; j < 0x1fff; j++){
            dmy += peek(j<<3,0);
            if (endflg != 0)    break;
        }
        if (endflg != 0)    break;
    }
    printf("%u¥n",i);
}
```

グラフィック LIO

拡張グラフィック機能(4096色中16色モード，GRCG／EGCを使用した高速描画等)はグラフLIOによりサポートされます．機種や動作状況によっては動作しないものもあります．

グラフLIOはN_{88}BASICで使用するために作られたプログラム・モジュールです．アセンブラやC言語等で使用する場合は，必ずしも使いやすいとはいいにくいのですが，グラフィックBIOSやGDC直接操作よりは，随分と良くなります．

◈ グラフLIOの使用方法

グラフLIO(以降，LIOと略す)は，BIOSのように割り込みによる呼び出しではなく，PC98本体内蔵のROM上にあるプログラム群です．17種のコマンドからできており，F9900H番地から呼び出しのためのエントリ・ポイント・オフセット・アドレスが17個分書かれていますので，これに従って呼び出します．使用する場合は，割り込みベクタ(ソフトウェア割り込み)にこのアドレスを書き込み，INT命令で呼び出すのが普通になっています．N_{88}BASICでは，AOH～AFH，CEHの割り込みを使用しますが，自由に変更できます(図4-21)．

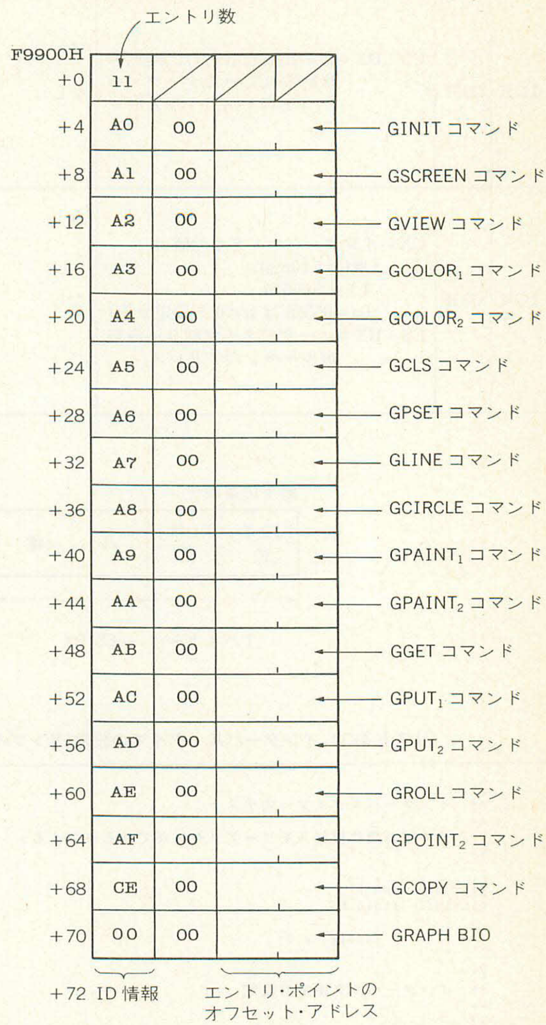
LIOを使用する前に，ワーク・エリア(UCW)と，スタック・エリアを確保しなくてはなりません．

UCWは1400Hバイト(GCOPYコマンドを使用しない場合は1200Hバイト)分必要です．セグメント・レジスタ「DS」でこのアドレスを指定します．オフセットは「0000H～13FFH」がUCWとして使用されます．

スタック・エリアはLIOが専用に128バイト使用します．レジスタ「SS：SP」で指定します．

必要に応じて，LIOのエント・ポイントを割り込みベクタにセットします．以下の説明では，N_{88}BASICと同様に，AOH～AFH，CEHの割り込みベクタを使用することを前提とします．

〈図4-21〉グラフLIOの使用方法



注1：各コマンドのエントリ・ポイントのセグメント・ベースは，F9900H(セグメント・レジスタへの格納値はF990H)
注2：N_{88} BASICシステムでの割り込みベクタ番号はA0～AF，CEを使用

◈ グラフLIOのコマンド

グラフLIOの入出力パラメーター一覧を図4-22に示します．

C言語によるグラフィック操作

グラフィック画面を扱うには，GDC直接操作やグラフィックBIOS，LIO等があります．GDCやBIOSの操作は，比較的難しく機能も低くなっています．LIOの操作は，比較的優しく機能も多くなっていますが，使用するにはアセンブラ・レベルの知識が多く必要になります．

最近では，コンパイラ言語が，安く手軽に使えるようになってきました．CPUの速度も高速になってきましたので，コンパイラ言語だけでプログラムを作成しても速度的に十分なことが多く，使用する頻度も上がってきています．

これらのコンパイラ言語には，独自にグラフィック操作命令を持つものも多くあります．そこで，C言語(Turbo C++　Ver.1.00)における，グラフィック関連の命令(関数)と，コンパイル時の注意について解説します．

〈図 4-22〉グラフ LIO の入出力パラメータ

割り込み番号	ルーチン名	機能	入力パラメータ	出力パラメータ
A0h	GINIT	初期化	なし	AH＝00h：正常終了
A1h	GSCREEN	モード設定	BX：パラメータ・リストへのポインタ（図a）	AH＝00h：正常終了，AH＝05h：不正呼び出し
A2h	GVIEW	ビューポート指定	BX：パラメータ・リストへのポインタ（図b）	AH＝00h：正常終了，AH＝05h：不正呼び出し
A3h	GCOLOR1	背景色指定	BX：パラメータ・リストへのポインタ（図c）	AH＝00h：正常終了
A4h	GCOLOR2	パレット番号と表示色の対応	BX：パラメータ・リストへのポインタ（図d）	AH＝00h：正常終了
A5h	GCLS	描画領域の塗りつぶし	なし	AH＝00h：正常終了
A6h	GPSET	点を打つ	BX：パラメータ・リストへのポインタ（図e） AH＝01h：フォア・グラウンド・パレット番号 AH＝02h：バック・グラウンド・パレット番号	AH＝00h：正常終了
A7h	GLINE	直線，方形を描く	BX：パラメータ・リストへのポインタ（図f）	AH＝00h：正常終了
A8h	GCIRCLE	円，楕円を描く	BX：パラメータ・リストへのポインタ（図g）	AH＝00h：正常終了，AH＝06h：演算オーバフロー
A9h	GPAINT1	色で塗りつぶし	BX：パラメータ・リストへのポインタ（図h）	AH＝00h：正常終了，AH＝05h：不正呼び出し AH＝07h：ワーク域不足
AAh	GPAINT2	タイルで塗りつぶし	BX：パラメータ・リストへのポインタ（図i）	AH＝00h：正常終了，AH＝05h：不正呼び出し AH＝07h：ワーク域不足
ABh	GGET	描画情報の格納	BX：パラメータ・リストへのポインタ（図j）	AH＝00h：正常終了，AH＝05h：不正呼び出し
ACh	GPUT1	描画情報の表示	BX：パラメータ・リストへのポインタ（図k）	AH＝00h：正常終了，AH＝05h：不正呼び出し
ADh	GPUT2	日本語の描画	BX：パラメータ・リストへのポインタ（図l）	AH＝00h：正常終了，AH＝05h：不正呼び出し
AEH	GROLL	描画画面の移動	BX：パラメータ・リストへのポインタ（図m）	
AFh	GPOINT2	ドットのパレット番号の取得	BX：パラメータ・リストへのポインタ（図n）	AH＝00h：正常終了，AL：パレット番号
CEh	GCOPY	ドット情報の格納	AX：左上点X座標 BX：左上点Y座標 CL：X方向ドット数 CH：Y方向ドット数 DI：バッファのオフセット・アドレス ES：バッファのセグメント・アドレス	AH：不定

(a) GSCREEN のパラメータ・フォーマット

BX＋0	＋1	＋2	＋3
画面モード	画面スイッチ	アクティブ・ページ	ディスプレイ・ページ

パラメータ	画面モード	画面スイッチ	アクティブ・ページ	ディスプレイ・ページ
00h	カラー・モード（640×200ドット）	表示あり	アクティブ・ページの番号（0～11）	ディスプレイ・ページの番号（0～31）
01h	モノクロ・モード（640×200ドット）	表示あり 高速書き込み		
02h	モノクロ・モード（640×400ドット）	表示なし		
03h	カラー・モード（640×400ドット）	表示なし 高速書き込み		
FFh	今までのモードを引き継ぐ			

(b) GVIEW のパラメータ・フォーマット

BX＋0 ＋1	＋2 ＋3	＋4 ＋5	＋6 ＋7	＋8	＋9
左上X座標（X_1）	左上Y座標（Y_1）	右下X座標（X_2）	右下Y座標（Y_2）	領域色	境界色

領域色：パレット番号(0～7)、塗りつぶししない(FFh)

境界色：パレット番号(0～7)、外枠を描かない(FFh)

(c) GCOLOR1 のパラメータ・フォーマット

BX＋0	＋1	＋2	＋3
未使用	バック・グラウンド・カラー（0～7）	ボーダ・カラー（0～7）	フォア・グラウンド・カラー（0～7）

(d) GCOLOR2 のパラメータ・フォーマット

BX＋0	＋2
パレット番号（0～7）	表示色コード（0～7）

(e) GPSET のパラメータ・フォーマット

BX＋0 ＋1	＋2 ＋3	＋4
X座標	Y座標	パレット番号（0～7）

(f) GLINE のパラメータ・フォーマット

BX +0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+10	+11	+12	+13
始点のX座標(X_1)		始点のY座標(Y_1)		終点のX座標(X_2)		終点のY座標(Y_2)		パレット番号1(0〜7)	描画指定 0:直　線 1:方　形 2:方　形 塗りつぶし	ライン・スタイル・スイッチ	パレット番号2 ライン・スタイル LOW	HI	タイル・パターン長

+14	+15	+16	+17
タイル・パターン格納域 オフセット・アドレス		セグメント・アドレス	

▶ライン・スタイル・スイッチ(+10)

- 00h：指定なし
- 01h：ライン・スタイルまたはパレット番号2指定あり
- 02h：タイル・パターン指定あり

▶パレット番号2(+11)

- 方形内部の塗りつぶし色指定
- 描画コード＝20hのみ有効

▶ライン・スタイル(+11, +12)

b_0 … b_7	b_8 … b_{15}
LO(+11)	HI(+12)

(g) GCIRCLE のパラメータ・フォーマット

BX +0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+10	+11	+12	+13
中心点X座標(CX)		中心点Y座標(CY)		X方向半径(RX)		Y方向半径(RY)		パレット番号1(0〜7)	フラグ	開始点X座標(SX)		開始点Y座標(SY)	

+14	+15	+16	+17	+18	+19	+20	+21	+22
終了点X座標(EX)		終了点Y座標(EY)		パレット番号2 or タイル・パターン長	タイル・パターン格納域 オフセット・アドレス		セグメント・アドレス	

▶フラグ(+9)

- b_0:開始点指示(0：なし　1：あり)
- b_1:開始線分指示(0：なし　1：あり)
- b_2:終了点指示(0：なし　1：あり)
- b_3:終了線分指示(0：なし　1：あり)
- b_4:描画方法(0：全楕円を描画, 1：1点のみ描画)
- b_5:塗りつぶし指示(0：なし　1：あり)
- b_6:タイル・パターン指示(0：なし　1：あり)

(h) GPAINT1 のパラメータ・フォーマット

BX +0	+1	+2	+3	+4	+5	+6	+7	+8	+9
開始点X座標(X)		開始点Y座標(Y)		領域色パレット番号	境界色パレット番号	ワーク域 最終オフセット		先頭オフセット	

▶ワーク域は16バイト以上

▶ワーク域はデータ・セグメント内にあること

(i) GPAINT2 のパラメータ・フォーマット

BX +0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+10	+11〜+15	+16	+17	+18	+19
開始点X座標(X)		開始点Y座標(Y)		未使用	タイル・パターン長さ(バイト)	タイル・パターン格納域 オフセット・アドレス		セグメント・アドレス		境界色パレット番号(0〜7)	未使用	ワーク域 最終オフセット		先頭オフセット	

ワーク域：図(h)参照

(j) GGET のパラメータ・フォーマット

BX +0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+10	+11	+12	+13
左上点X座標(X1)		左上点Y座標(Y1)		右下点X座標(X2)		右下点Y座標(Y2)		格納域 オフセット・アドレス		セグメント・アドレス		格納域長さ	

(k) GPUT1 のパラメータ・フォーマット

BX +0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+10	+11	+12	+13
左上点 X座標(X)		左上点 Y座標(Y)		格納域 オフセット・アドレス		格納域 セグメント・アドレス		格納域 長さ(バイト)		描画 モード	カラー・スイッチ	フォア・グラウンド・カラー (0~7)	バック・グラウンド・カラー (0~7)

▶描画モード(+10)
指定領域上のパターンと
格納域パターンとの論理演算
00h:T
01h:NOT
02h:OR
03h:AND
04h:XOR

(m) GROLL のパラメータ・フォーマット

BX+0	+1	+2	+3	+4
上下ドット数 (−399~399)		左右ドット数 (−639~639)		フラグ

▶フラグ(+4)
00h:移動後の残り領域を
　　パレット0にする
01h:移動後の残り領域を
　　バック・グラウンド・
　　カラーにする

(l) GPUT2 のパラメータ・フォーマット

BX+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
左上 X座標(X)		左上 Y座標(Y)		日本語JIS コード		描画 モード	カラー・スイッチ	フォア・グラウンド・カラー (0~7)	バック・グラウンド・カラー (0~7)

▶描画モード(+6)
指定領域上のパターンと
格納域パターンとの論理演算
00h: T
01h: NOT
02h: OR
03h: AND
04h: XOR

(n) GPOINT2 のパラメータ・フォーマット

BX+0	+1	+2	+3
X座標,(X)		Y座標,(Y)	

◈ Turbo C++のグラフィックについて

Turbo C++では，70を超える高機能なグラフィック関数が用意されており，仮想画面，ライン・スタイル，塗りつぶしパターンの設定等多くの機能がサポートされています．これらの機能はグラフィック・ドライバによるものです．

グラフィックス・ドライバには，ノーマル(640×400)用のPC98. BGIとハイレゾ(1120×750)用のPC98HI. BGIの二つがあります．これらのグラフィックス・ドライバはハードウェアをチェックし，最適なハードウェアを使用することが可能です．

◈ グラフィックス・ドライバの使い方

グラフィックス・ドライバは，プログラム起動後ロードして使う方法と実行ファイルに組み込んで使う方法の2通りの方法があります．グラフィックス・ドライバは巨大ですので，ロードする方法は実行ファイルのサイズが小さくなりますし，コンパイル時間も短くなります．しかし，外部にドライバを必要とするのは面倒な面も多いと思います．開発中はロードして使い完成したら組み込んで置く等，自分の使い方に応じて使い分けると良いでしょう．

● ロードして使う方法

プログラムの最初でinitgraphでロードするグラフィックス・ドライバ(*. BGI)のディレクトリを，次に示す例のように指定します．

```
initgraph(&graphdriver,&graphmode,"a:
¥¥turboc¥¥bgi¥¥");
```

グラフィック関数を使用する場合，graphics.libをリンクする必要がありますので，次のようにコンパイルします．

tcc ソース・ファイル名 graphics.lib

● 組み込んで使う方法

まず，グラフィックス・ドライバ(*.BGI)をBGIOBJ.EXEを使ってOBJファイルに変換します．

例　bgiobj pc98
　　pc98.bgi を pc98.obj に変換する．拡張子は付けない．

後は，コンパイルのときに次のように作成されたOBJファイルを指定すれば，組み込むことができます．

tcc ソース・ファイル名 graphics.lib OBJファイル名

〈リスト 4-2〉グラフィックのテスト・プログラム

```
/*
**  グラフィックのテスト
**
**  ドライバをロードする場合
**      tcc sample.c graphics.lib
**  ドライバを組み込む場合
**      tcc -DDRIVER_LINK sample.c graphics.lib pc98.obj
*/

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <graphics.h>

void grapherrexit(int errorcode);   /*  ｸﾞﾗﾌｨｸｽｴﾗｰ終了  */

int main()
{
    int     gdriver = DETECT;       /*  自動検出要求     */
    int     gmode,errcode;

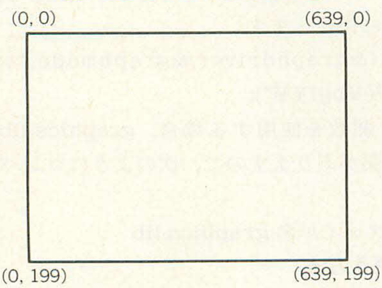
#ifdef  DRIVER_LINK                 /*  ドライバを組み込む方法     */
    if ((errcode = registerbgidriver(PC98_driver)) < 0) grapherrexit(errcode);
    initgraph(&gdriver,&gmode,"");
#else                               /*  ドライバをロードする方法   */
    initgraph(&gdriver,&gmode,"a:¥¥turboc¥¥bgi¥¥");
#endif
    if ((errcode = graphresult()) < 0)  grapherrexit(errcode);

    setcolor(WHITE);
    setfillstyle(CLOSE_DOT_FILL,LIGHTBLUE);
    bar(50,50,590,350);
    setfillstyle(XHATCH_FILL,LIGHTGREEN);
    pieslice(320,200,45,315,100);

    getch();                        /*  何かキーを押したら終了  */
    closegraph();
    return 0;
}

/*
**  グラフィックエラー終了
*/
void grapherrexit(int errcode)
{
    printf("Graphics error : %d¥n",errcode);
    exit(1);
}
```

〈図 4-23〉グラフィックの座標



〈図 4-24〉カラー・コードと色

color	値	実際の色	color	値	実際の色
BLACK	0	黒	DARKGRAY	8	暗い灰色
BLUE	1	青	LIGHTBLUE	9	明るい青
GREEN	2	緑	LIGHTGREEN	10	明るい緑
CYAN	3	水色	LIGHTCYAN	11	明るい水色
RED	4	赤	LIGHTRED	12	明るい赤
MAGENTA	5	紫	LIGHTMAGENTA	13	明るい紫
BROWN	6	暗い黄色	YELLOW	14	黄色
LIGHTGRAY	7	灰色	WHITE	15	白

OBJ ファイルがカレント・ディレクトリにない場合はフルパス指定します．

プログラム中では，initgraph の前に registerbgidriver でドライバをシステムに登録する必要があります．

例　registerbgidriver(PC98_driver);リンクされているノーマル用ドライバを登録

サンプル・プログラムでは，二つの方法でコンパイルできるように，DRIVER_LINK が定義されているかどうかで条件コンパイルするようになっています．

サンプル・プログラム SAMPLE.C をリスト 4-2 に示します．

ロードして使う場合には，そのままコンパイルします．

tcc sample.c graphics.lib

組み込んで使う場合には，−D オプションを使用し

〈図 4-25〉 エラー・コード

エラー・コード	値	意味
grOK	0	エラーなし
grNoInitGraph	−1	(BGI)グラフィックスがインストールされていない
grNotDetected	−2	グラフィックス・ハードウェアが検出不能である
grFileNotFound	−3	デバイス・ドライバ・ファイルが見つからない
grInvalidDriver	−4	不正なデバイス・ドライバ・ファイル
grNoLoadMem	−5	ドライバをロードするメモリが不足
grNoScanMem	−6	塗りつぶしスキャンでメモリが不足
grNoFloodMem	−7	領域塗りつぶしでメモリが不足
grFontNotFound	−8	フォント・ファイルが見つからない
grNoFontMem	−9	フォントをロードするメモリが不足
grInvalidMode	−10	選択したドライバに対する不正なグラフィックス・モード
grError	−11	グラフィックス・エラー
grIOerror	−12	グラフィックスI/Oエラー
grInvalidFont	−13	不正なフォント・ファイル
grInvalidFontNum	−14	不正なフォント番号
grInvalidVersion	−18	ファイルのバージョンが不正

てDRIVER_LINKを定義してコンパイルします．

```
tcc -DDRIVER_LINK sample.c graphics.lib pc98.obj
```

◈ グラフィック関数

70以上もあるグラフィック関数すべてを説明するわけにはいきませんので，グラフィックの機能を使用するのに最低限必要と思われるものを図4-26にまとめました．

● 座標

座標は画面左上を(0，0)としてx座標は左から右に向かって増加し，y座標は上から下に向かって増加します．グラフィックの座標を図4-23に示します．

● 色

カラー・コードと色の関係は図4-24のとおりです．98のカラー・コードと異なるのでシンボルを使用しない場合には注意が必要です．8色モード時に8〜15を指定した場合，0〜7に変換されます．

エラー・コード一覧を図4-25に，またTurbo C++のグラフィック関連関数のリファレンスを図4-26に示します．

◈参考文献◈

(1) PC98シリーズテクニカルデータブック HARDWARE編，㈱アスキー，1993年10月25日．

(2) PC98シリーズテクニカルデータブック BIOS編，㈱アスキー，1993年4月15日．

(3) EPSON PCシステム・ガイド，㈱クリエイト・クルーズ，平成5年3月31日．

(4) 浅野泰之，壁谷正洋，金磯善博，萊野雅彦；PC-9801システム解析（下），㈱アスキー，1983年12月1日．

(5) トランジスタ技術スペシャル No.3，CQ出版㈱，昭和62年5月1日．

(6) 海原系；PCのハードウェアを理解する，別冊インターフェース BootStrap Project-2 No.4，CQ出版㈱，1993年7月1日．

(7) NEC電子デバイス アプリケーション・ノート μPD71037/μPD72020DMAコントローラ，日本電気㈱．

(8) μPD7220GDCユーザーズ・マニュアル，日本電気㈱，December 1987．

(9) NEC電子デバイス ユーザーズ・マニュアル μPD71059/μPD71037/μPD71055/μPD72020/μPD7220A/μPD71054，日本電気㈱．

好評発売中！　ディジタル・オーディオの本です

トランジスタ技術 SPECIAL No.21

最もポピュラーな最新技術を理解しよう

特集 ディジタル・オーディオ技術の基礎と応用

B5判 160ページ

定価1,540円(税込)

CQ出版社

Contents

ディジタル・オーディオの基礎／ディジタル・フィルタの基礎／ディジタル・オーディオ・インターフェースの基礎／ΔΣ変調方式D-Aコンバータの基礎／マルチ・エフェクタとサラウンド・プロセッサの製作／DAIボードの製作／ディジタル・プリアンプの製作／LSIキットを使ったCDシステム／光磁気ディスク・オーディオ・ファイル装置

〈図 4-26〉 Turbo C++のグラフ関数

● グラフィックス・システムを初期化する

関数/引き数
initgraph(graphdriver, graphmode, pathdriver);
解　　説

グラフィックス・ドライバをロードまたは登録ずみのドライバを有効にしてグラフィック・システムを初期化する.

graphdriver	値	意　　味
DETECT	0	自動検出を要求する
PC98	1	ノーマル・モードで使用する
PC98HI	2	ハイレゾ・モードで使用する

graphmode	値	意　　味
PC98C8	0	8色モードで使用する
PC98C16	1	16色モードで使用する

pathdriver　グラフィックス・ドライバをロードするときに検索するディレクトリ名で, 存在しなかったときはカレント・ディレクトリを検索する.

graphdriver に DETECT を指定した場合, ハードウェアを自動検出して動作する. エラーが発生しなかった場合, 以下の値が代入される.

(graphdriver,graphmode は INT へのポインタ)

graphdriver	値	意　　味
PC98	1	ノーマル・モード
PC98HI	2	ハイレゾ・モード

graphmode	値	意　　味
PC98C8	0	8色モード
PC98C16	1	16色モード

● グラフィックス・システムの使用を終了する

関数／引き数
closegraph();
解　　説
グラフィックス・システムが割り当てたすべてのメモリを解放し, 画面モードを initgraph が呼ばれる前の状態にする.

● ユーザがリンクしたグラフィックス・ドライバをシステムに登録する

関数／引き数
registerbgidriver(driver);
解　　説

実行ファイルにグラフィックス・ドライバを組み込んだ場合 initgraph の前にこの関数でシステムに登録する必要がある. エラーが発生した場合, 負のグラフィックス・エラー・コードを返す. エラー・コードは図 4-25 参照.

driver	リンクされているドライバ
PC98-driver	PC98.BGI
PC98HI-driver	PC98HI.BGI

● 最後に発生したグラフィックス操作に対するエラーのエラー・コードを返す

関数／引き数
graphresult();
解　　説
エラー・コードは図 4-25 参照.

● グラフィックス画面を消去する

関数／引き数
cleardevice();

● 描画色を設定する

関数／引き数
setcolor(color);
解　　説
color は図 4-24 を参照

● 塗りつぶしパターンと色を設定する

関数／引き数
setfillstyle(pattern,color);
解　　説

塗りつぶしの色は setcolor では変更できない.

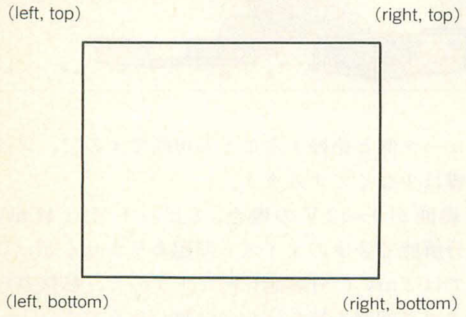
▶ pattern

EMPTY_FILL	0	背景色で塗る
SOLID_FILL	1	ベタ塗り
LINE_FILL	2	横線で塗る
LTSLASH_FILL	3	／／／で塗る
SLASH_FILL	4	／／／(太)で塗る
BKSLASH_FILL	5	＼＼＼(太)で塗る
LTBKSLASH_FILL	6	＼＼＼で塗る
HATCH_FILL	7	＋＋＋で塗る
XHATCH_FILL	8	×××で塗る
INTERLEAVE_FILL	9	インターリーブ線で塗る
WIDE_DOT_FILL	10	点(間隔広)で塗る
CLOSE_DOT_FILL	11	点(間隔狭)で塗る

color は図 4-24 を参照.

● 長方形を描く

関数／引き数
rectangle(left,top,right,bottom);
解　　説
図のような，座標の長方形を描く．

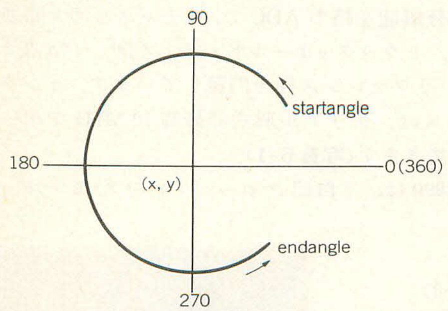


● 長方形(塗りつぶし)を描く

関数／引き数
bar(left,top,right,bottom);
解　　説
パラメータについては rectangle を参照

● 円弧を描く

関数／引き数
arc(x,y,startangle,endangle,radius);
解　　説
中心座標(x,y)，半径ドット数 radius，描画開始角度 startangle，終了角度 endangle，反時計回りに円弧を描く．startangle，endangle は，時計の3時の位置が0，12時の位置が90で，startangle は endangle より小さくなるように設定する．

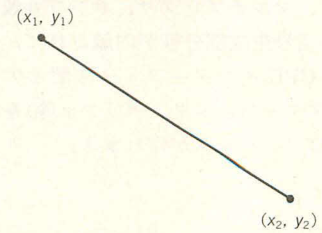


● 円弧を描きその中を塗りつぶす

関数／引き数
pieslice(x,y,startangle,endangle,radius);
解　　説
パラメータについては arc を参照

● 2点間を結ぶ直線を描く

関数／引き数
line(x1,y1,x2,y2);
解　　説
2点(x1,y1)と(x2,y2)を結ぶ直線を描く．



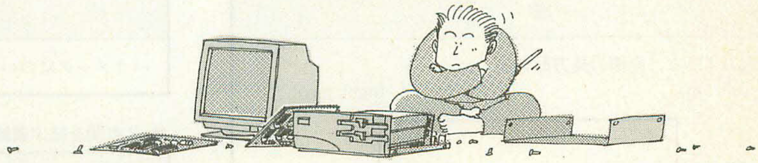
● 指定座標に点を表示する

関数／引き数
putpixel(x,y,color);
解　　説
(x,y)に color の色の点を表示する．color は図4-24参照

● 指定座標に文字列をグラフィック画面に描く

関数／引き数
outtextxy(x,y,text);
解　　説
x,y 書き始め座標．text 描画する文字列

5　AD7890-4を使った拡張基板の製作



A-Dコンバータの製作

最近のA-Dコンバータは，多機能，高性能になり，ワンチップで，マルチプレクサ，サンプル&ホールド，その他の制御信号生成部分等が内蔵されて，あとは必要に応じた，CPUインターフェース部とアナログ部(アンチエリアス・フィルタ，バッファ等)を外付けするだけでA-Dコンバータが作れます．

◈ 目標

単電源のOPアンプを使用する機会が多いので，測定範囲としては，0〜12Vを中心に，10kHz程度の波形を観測することを目標にしました．サンプリング周期は10μs程度，分解能は8〜12ビット程度を目標とします．

PC9801等に接続するには，パラレル出力タイプのA-Dコンバータ(以下ADCと略す)のほうがインターフェースが簡単に作れて便利です．反面，ノイズだらけのコンピュータ内部の拡張基板上に，ADCやアナログ関係の回路を載せなくてはならないために，ノイズ対策等，実装面での配慮が必要になります．

シリアル出力のADCですと，ADC部分とインターフェース部分を比較的少ない本数の制御線でつなぐだけですので，拡張基板から遠ざけることも簡単にできます．また，間にフォト・カプラ等を付けることで，コンピュータ側と絶縁することも可能ですので，ノイズの影響は少なくてすみます．

測定範囲が0〜12Vの場合，8ビットでは47mV程度の分解能で多少のノイズも問題ありませんが，12ビットでは3mVの分解能を持ちますので，拡張基板上にアナログ回路を持ち込むのは難しそうです．

◈ パラレル・インターフェースとシリアル・インターフェース

ADCが変換したデータをCPUへ送るためのインターフェースとして，パラレル・インターフェースとシリアル・インターフェースの2種類があります．

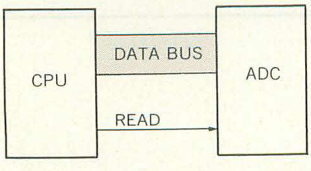
図5-1のパラレル・インターフェースは，CPUへパラレルで変換データを送ります．一般のCPU等に接続することを前提に作られていますので，通常のCPU周辺LSIと同様に，I/Oポートとして動作し，データ・バスを繋ぐだけの簡単な設計で作れます．

いっぽう図5-2に示すシリアル・インターフェースは，DSPやワンチップCPU等の，拡張用バスを持たないプロセッサに向いていて，プロセッサ内部にシリアル入力ポートを持っていれば，少ない結線数で接続できます．

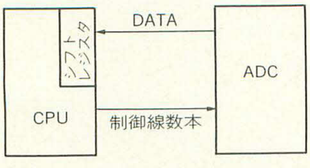
◈ AD7890について

今回使用したAD7890は，変換時間5.9μs，12ビットの分解能を持つADCで，8チャネルのマルチプレクサ，トラック・ホールド・アンプ(サンプル&ホールド)，リファレンス等を内蔵しています．インターフェースは，シリアル形式で最高10MHzでデータを転送できます(写真5-1)．

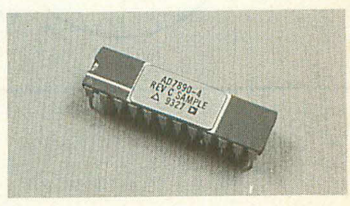
AD7890は，「自己クロック(マスタ)モード」と



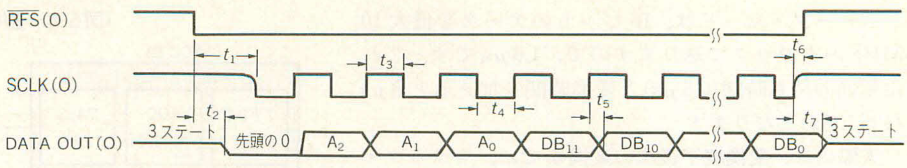
〈図5-1〉
パラレル・インタースェースのADC



〈図5-2〉
シリアル・インターフェースのADC



〈写真5-1〉
AD7890



〈図5-3〉
自己クロック(マスタ)モードの出力レジスタの読み出しタイミング

「外部クロック(スレーブ)モード」を持ちます．

マスタ・モードは，データ転送のための信号を自分自身が発生するので比較的簡単にインターフェース回路が設計できます．ADCに与える信号は変換開始信号だけになります．クロックには，ADCが変換用クロックに使用する2.5 MHzを使用するため，転送速度がやや遅くなってしまうのが欠点です．実質的な最高サンプリング周波数は78 kHzになってしまいます．

スレーブ・モードは，外部からデータ転送に必要な制御信号をADCへ与えて動作させるモードです．多くの制御信号を与えなくてはならないので，インターフェースをロジック回路で作ると設計が大変です．しかし，変換用クロックとは別に，データ転送用のクロックを与えられ，そのクロック・タイミングを自由に変化させられるので，マイクロプロセッサやDSPからソフトウェア制御をするには向いています．また，データ転送用クロックに最高10 MHzまで使用できるために，データ転送時間が少なくて済みます．実質的なサンプリング周波数は117 kHz(127 kHz)まで可能です．

◈ AD7890のタイミング

送出されるデータは図5-3のように，変換データD_0～D_{11}の12ビットに加え，チャネル情報がA_0～A_2が3ビットに，先頭のダミー・ビットが加わり全部で16ビット構成です．

また，ADCに対して，シリアルのデータ(コマンド)を送ることで，「使用チャネル指定(3ビット)，ソフトウェアに変換開始(1ビット)，スタンバイ・モード(1ビット)」の指示も可能です(図5-4)．

今回はソフトウェアからの，変換開始スタンバイ・モード指示は行いませんでした．

マスタ・モードでは，16ビットのデータを2.5 MHzのクロックで送りますので，6.4 μsかかります．この

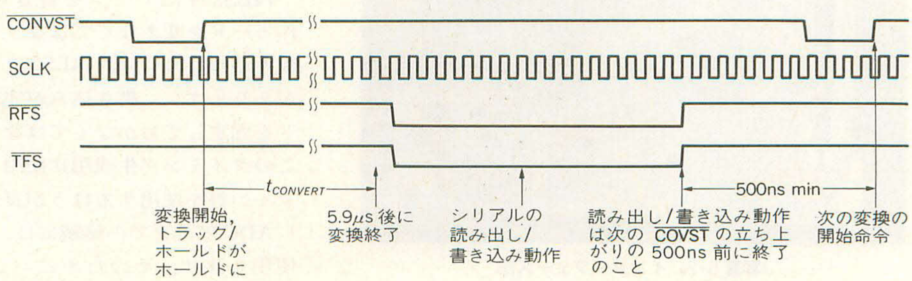
〈図5-4〉制御レジスタのビット・アサイン

MSB

A_2	A_1	A_0	CONV	STBY

ビット	説明
A_2	アドレス入力．この入力はマルチプレクサのチャネル選択アドレスの最上位ビット(MSB)
A_1	アドレス入力．この入力はマルチプレクサのチャネル選択アドレスの上位2番目のビット
A_0	アドレス入力．この入力はマルチプレクサのチャネル選択アドレスの最下位ビット(LSB)．アドレスが制御レジスタに書き込みされると内部パルスが発生する．この内部パルス幅はC_{EXT}端子に接続されたコンデンサの容量値によって決まる．この内部パルスがアクティブ期間中には変換動作は開始されない．このことにより，マルチプレクサのセトリング時間とトラック・ホールドのアクイジション時間だけトラック・ホールドがホールド・モードになり，変換の開始がウェイトされる．またMUX OUT端子とSHA IN端子の間にアンチエリアシング・フィルタを接続する応用では，SHA INに加わった信号がサンプリングされるタイミングに，フィルタのセトリング時間も加えることができる．内部パルスがタイムアウトするとトラック・ホールドがホールド・モードとなり，変換が開始される．
CONV	変換開始．このビットに1を書き込むと，CONV ST入力と同様に変換が開始される．このビットが1の場合でも，連続的な変換動作は行われない．このビットが1の場合には書き込み動作の6番目のシリアル・クロック・サイクル後に内部パルスと変換が開始する．このビットが1の場合にはハードウェアによる変換開始入力($\overline{\text{CONVST}}$)は禁止される．このビットに0を書き込むと$\overline{\text{CONVST}}$入力がイネーブルされる．
STBY	スタンバイ・モード入力．このビットに1を書き込むと，デバイスはスタンバイ(パワーダウン)モードになる．このビットに0を書き込むと，デバイスは通常の動作モードとなる．デバイスはSCLKの7番目の立ち上がりエッジまでスタンバイ・モードに入らない．このため，デバイスをスタンバイ・モードにする場合にはシリアル書き込み動作で7個のシリアル・クロック・パルスが必要

ため，変換時間と合わせて12.3 μs必要になります．さらに，データの読み出し終了直後から，0.5 μs以内は，次の変換を開始してはいけませんので(これが守られないと，AD7890の性能を劣化させることがある)，合計12.8 μs(78 kHz)となります．



〈図5-5〉
外部クロック(スレーブ)モードで最適性能を得る動作タイミング

スレーブ・モードは，16 ビットのデータを最大 10 MHz のクロックで送りますので，1.6 μs です．これに変換の待ち時間(0.5 μs)と変換時間を加えると 8 μs (125 kHz)となります．

実際には，変換終了後次の変換までに，トラック・ホールドのアクイジション時間(トラック・ホールドが確定する時間)の 2 μs 間おかなくてはなりませんが，先のデータ転送時間のほうが長くなるので問題ありません．また，図 5-5 に示すように変換データの転送中に，同時にマルチプレクサのチャネルを変更する場合は，このチャネル変更終了後から，アクイジション時間を置かなければなりません．この場合は，チャネル変更に SCLK が 6 クロック(10 MHz ならば 0.6 μs)かかりますので，2 μs＋0.6 μs＋変換時間で，8.5 μs (117 kHz)となります．

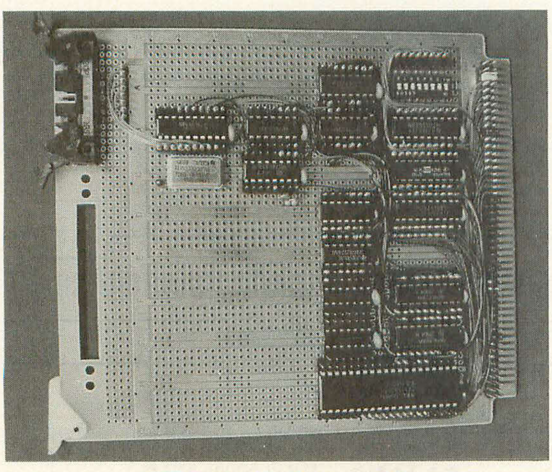
◈ 全体の構成

全体の構成は図 5-6 のようになっています．拡張基板は CPU インターフェース部分のみで，ADC 側とはフラット・ケーブル(シールド・ケーブルが良い)で接続します．ADC のインターフェースにはフォト・カプラを入れてノイズ対策すると良いのですが，今回は電源を分離するだけにしました．

図 5-7 (a)，(b)に全回路を示します．

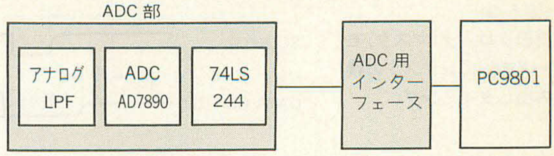
インターフェースは PC98 の拡張スロットに入れるために，ロジック(TTL)で組みますので設計が楽な「マスタ・モード」を使用しました．変換速度が落ちますが，目的の 10 kHz 程度の波形観測はできそうです．

ADC からの変換データは 100 kHz 近い転送レートで行われます．速度の遅い CPU ですと，DMA 転送しないと間に合わない速度です．しかし，ADC のデータが 12(16)ビットであるために，ノーマル・モードの PC98 では，一度に DMA 転送できません．また，2 回に分けると，タイミング生成が難しくなります．



〈写真 5-2〉 インターフェース部

〈図 5-6〉 全体の構成



また，DMA 転送ですと，一度に 64K バイトまでのデータしか送れないために，長時間のサンプリングも難しくなります．

今回は，ある程度高速な CPU を前提として，タイマ割り込みでサンプリングしています．割り込み処理は，とても時間がかかるのですが，遅い CPU でも，割り込み間隔を延ばすことで，自由にサンプリン・グレートを遅くして対応できます．

◈ インターフェース部分

シリアルとパラレルの変換部分には，シフトレジスタの 74LS299(U5，U6)を使用します．これ一つだけでシリアル 8 ビットの受信と送信ができますので，二つ直列にすることで 16 ビットのデータを扱います．

制御信号のタイミング信号は，ADC 側から供給されるので，インターフェース部分は比較的簡単にできますが，読み取りのソフトウェア処理の負担を軽くするために，いくつか付加回路を付けました．

(1) 変換データを読み出すと同時に，次のサンプリング開始信号が自動的に送出されます．74LS74(U3A)を使用し，データ読み出しで CONVST を“H”(変換開始)に，RFS が“H”(変換データ送出開始)になったら，CONVST を“L”にします．

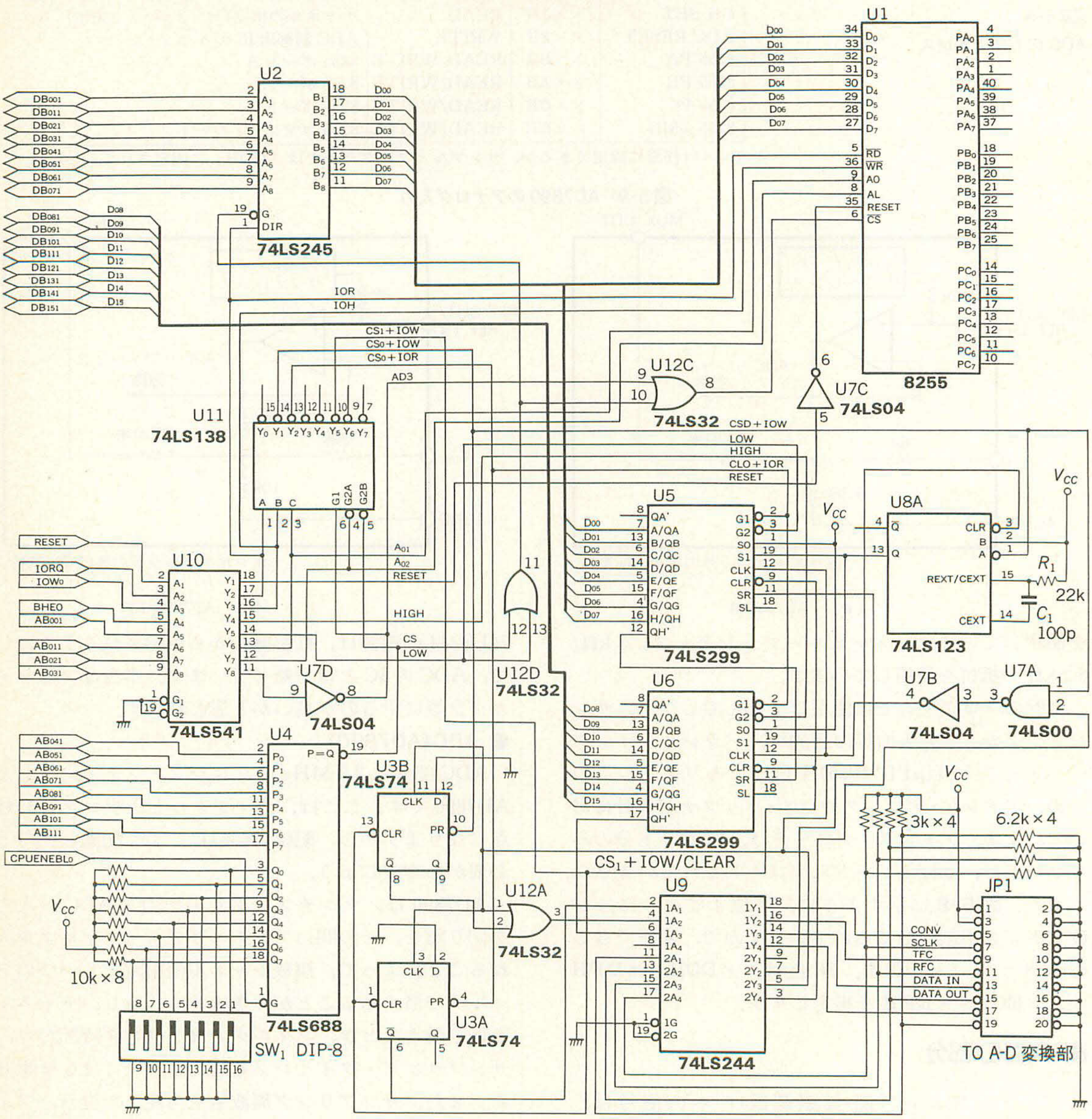
(2) ADC へのコマンド送出も，送りたいデータを書き込むだけで，自動的に送出制御信号を作ります．この場合も，74LS74(U3B)を使用し，データ書き込みで TFS(コマンド送出開始)を“L”にします．RFS が“H”になったら，同時に TFS も“H”にします．

(3) コマンドの送出は，74LS299 に送出コマンドを書き込み，A-D 変換データ読み込みと同時に，自動的にコマンド送出をしています．受信のために送られてきた SCLK のクロックで 74LS299 に書かれたデータがシリアル変換されて出ていきます．もちろん，入れ替わりで A-D 変換データが 74LS299 へ入ってきます．

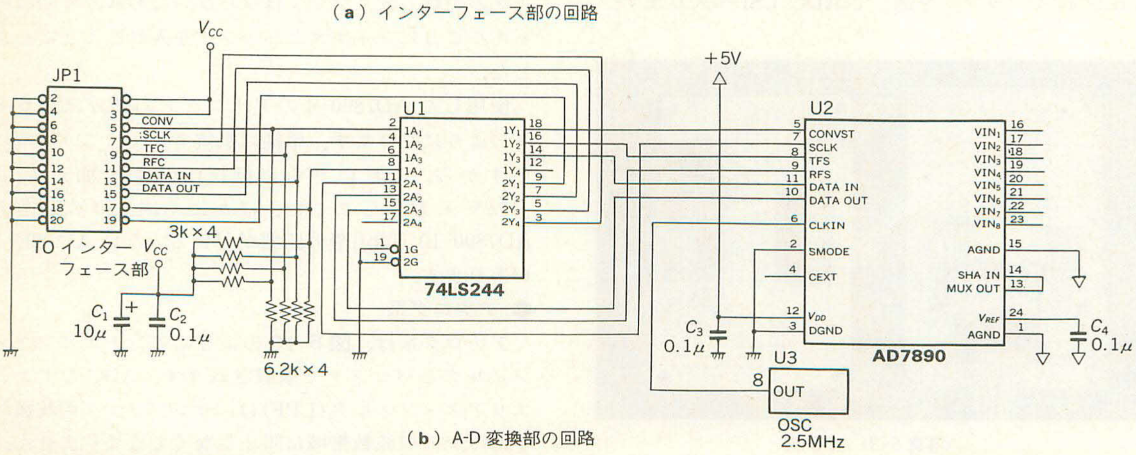
74LS299 にデータを書き込むには，74LS299 の動作モードを変えなくてはなりません．また，データを書き込むためには，74LS299 にクロックを与える必要がありますし，書き込みが完全に終了するまで，モードを固定しておかなくてはなりません．74SL123 は，このタイミング生成用に使用しましたが，ディレイ・ライン等を使用するほうが良いかもしれません．

ADC 部分までの接続には，フラット・ケーブル等を使用します．そのためにバッファ用として 74LS244

〈図 5-7〉全体の回路



（a）インターフェース部の回路



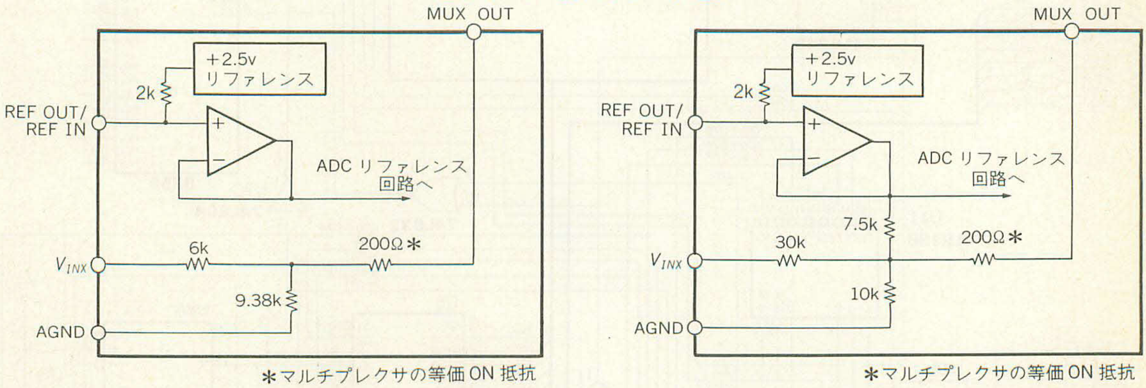
（b）A-D 変換部の回路

〈図5-8〉
ADCのI/Oアドレス

DATA READ	××0H	READ	変換されたデータの読み出し(16ビット幅)
CH SET	××1H	READ	チャネルの指定(ビット5~7を使用)
ADC RESET	××2H	WRITE	ADC制御用ICのリセット
8255 PA	××8H	READ/WRITE	8255ポートA
8255 PB	××AH	READ/WRITE	8255ポートB
8255 PC	××CH	READ/WRITE	8255ポートC
8255 CMD	××EH	READ/WRITE	8255コマンド・ワード

(××は任意に設定できるが，サンプル・プログラムでは「DDH」に固定されている)

〈図5-9〉AD7890のアナログ入力



(a) AD7890-4　　(b) AD7890-10

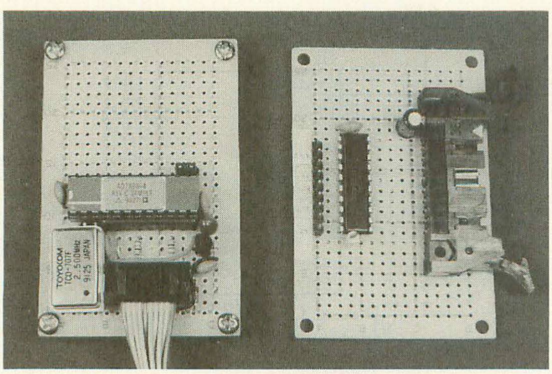
を使用しています．ターミネーションとして，3kΩ/6.2kΩの抵抗を使用しています．

今回のプログラムでは使用しませんでしたが，サンプリングのトリガ入力等の目的で，パラレル・インターフェースLSI(μPD71055)を付けてあります．このため，アドレス・デコーダやバス・バッファが余計に付いています．アドレス・デコーダは，12ビット分のみデコードし，上位4ビットはデコードされていません．

また，図5-8に示すように，下位4ビットにはI/Oアドレスが割り当てられていますので，変更できるのは8ビット分です．普通は，×D0H～×DFHか，×E0H～×EFHを指定します．

◈ ADC部分

インターフェース部(拡張基板)からの信号は，74LS244をバッファを通してADC-LSIへ入ります．



〈写真5-3〉ADC部

74LS244の電源は，拡張基板から供給されるようにして，ADCのICとは分離しています．本当はフォト・カプラ等にするのが良いかと思われます．

● ADC(AD7890)

ADC部分は2.5MHzのクロック・ジェネレータとAD7890です．ここは，ディジタルとアナログの分岐点になりますから，配線(実装)は，よく配慮して行う必要があるでしょう．

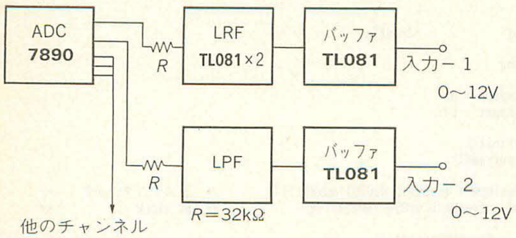
AD7890は，マルチプレクサの出力とADCの入力を切り離せ，その間にアンチエリアス・フィルタを入れることによって，複数チャネル使用時でも一つのフィルタで済ませることができます．しかし，チャネルを切り替えたとき，フィルタのセトリング時間だけ，サンプリング・タイミングを延ばさなくてはなりません．また，サンプリング周波数を可変した場合，フィルタの定数も変えなくてはならないために，今回はチャネルごとにフィルタとバッファを入れることにしました．

使用したAD7890-4のアナログ入力部分は図5-9(a)のようになります．単純に抵抗で分圧しているだけですから，これにさらに抵抗(32kΩ)を加えて，0～12V入力としました．ほかにも，±10V入力のAD7890-10[図(b)]や分圧器が付いていないAD7890-2があります．

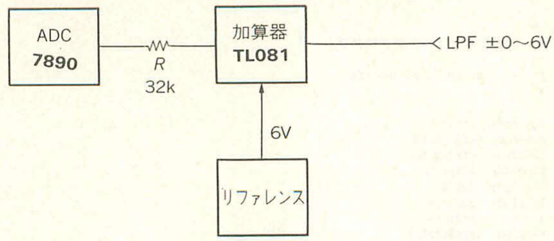
● アナログ系

アナログ系は，図5-10のようにアンチエリアス・フィルタとバッファで構成されます．ADCのアンチエリアス・フィルタ(LPF)は，サンプリング周波数の半分以上の周波数帯域は阻止しなくてはなりません．

〈図 5-10〉 アナログ系の構成

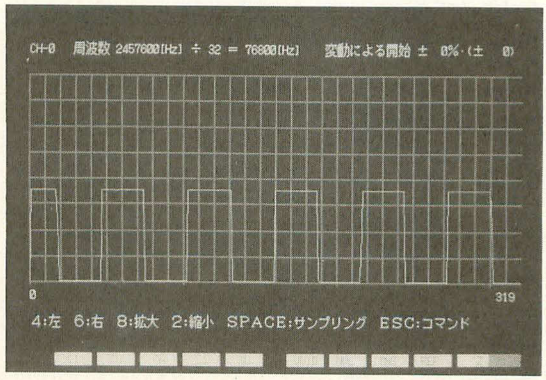


〈図 5-11〉 マイナス電圧が必要なときの構成

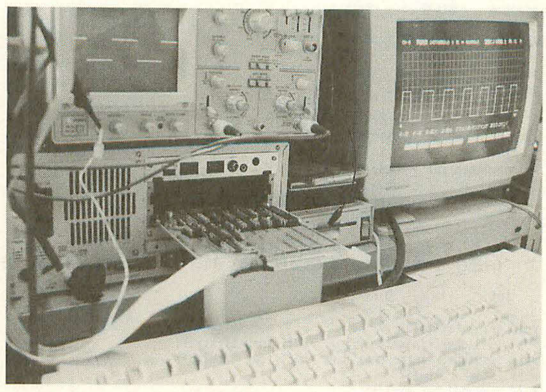


〈図 5-12〉 ADC サンプリング・プログラム

コマンド名	内　　容
SPACE	サンプリング開始
4/6	サンプリングされたデータの左右スクロール表示
2/8	サンプリングされたデータの拡大・縮小(時間軸)
L	保存されたデータ読み出し(FILE)
S	サンプリングされたデータの保存(FILE)
P	自動トリガのためのデータの変動許容率(%)
R	サンプリング時間設定(8253 へのデータ)
J	サンプリングされたデータの指定位置の表示
C	マルチプレクサのチャネル切り替え
Q	プログラムの終了



〈写真 5-4〉 ADC サンプリング・プログラムによる表示



〈写真 5-5〉 測定風景

分解能(ビット数)が多いほど，その条件は厳しくなってきます．

バッファは，入力インピーダンスを高くするためのもので，OP アンプを非反転増幅器のボルテージ・フォロワ(増幅率＝1)で使用します．OP アンプには TL081 を使用しました．

0～12 V 入力の場合は問題ありませんが，マイナスの電圧も必要な場合は，AD7890-10 を使用するか，アナログ回路中にサム・アンプ(加算器)を付けてバイアスをかけます．図 5-11 のようにすれば，いろいろなレンジで使用することもできます．

◈ ADC のソフトウェア

ADC を操作し，データのサンプリング，データの表示のためのプログラムが必要です．

PC9801 のタイマ LSI(8253)のチャネル#0 を使ったインターバル・タイマで，データのサンプリングを行います．CPU の割り込み処理速度によって調整する必要がありますが，最大 76.8 kHz になります．

8253 のクロックによって，8253 に与える分周比が変わってきます．システム・クロックが 10 MHz 系では，8253 のクロックは 2.4576 MHz となり，76.8 kHz のサンプリング・レートを出すには「32」となります．システム・クロックが 8 MHz 系ならば，19.968 MHz のクロックですから「26」となります．

A-D 変換は，DATA READ のリードを行った直後に，自動的に行われます．したがって，ADC RESET のライト後，最低でも 1 回，DATA READ をダミー・リードしなくてはなりません．

また，ADC のマルチプレクサのチャネルを変えるには，CH SET に変更したいデータを書き込みます．チャネルは 8 個ですから，3 ビット分のデータをビット 5～7 へ入れます．ビット 0～4 までは“0”にしておきます．このチャネル設定は，DATA READ 後，変換時間以内(5.9 μs)に行わないとなりません．

◈ ADC サンプル・プログラム

今回の実験のために作成したプログラムのリスト ADC.C　ADCSUB.ASM を紹介します．ADCSUB.ASM は ADC.C のコンパイル，リンク時に組み込みます．

使用したコンパイラは，Turbo C++ V.1.00，および Turbo ASSEMBLER です(写真 5-4)．

〈リスト 5-1〉ADC サンプリング・プログラム

```c
/*
**  ＡＤＣサンプリングプログラム
**
**  tcc -mc adc.c adcsub.asm
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <alloc.h>
#include <dos.h>
#include <conio.h>
#include <pc98.h>
#include <sys¥stat.h>
#include <io.h>
#include <fcntl.h>

#define     MAX_DATA        655361
#define     MAX_INT         32767

#define     TIMER_VEC   0x08
#define     TIMER_0     0x71
#define     TIMER_MODE  0x77
#define     PIC_IMR     0x02
#define     PIC_OCW1    0x02
#define     PIC_OCW2    0x00
#define     EOI         0x20

#define     GRPH_X_MAX  639
#define     GRPH_Y_MAX  399
#define     GRPH_Y_OFF  40

#define     ESC         0x1b

#define     GDC_STAT    0xa0
#define     GDC_CMD     0xa2
#define     GDC_PARA    0xa0

#define     VECTW       0x4c
#define     VECTE       0x6c
#define     CSRW        0x49
#define     TEXTW       0x78
#define     WRITE       0x20
#define     START       0x6b

#define     BIOS_CRT    0x18

#define     MAX_X_BAI   10
#define     DATA_RANGE  255

    int             ssd = 0,    ssp = 0;
    int             sysclock;
    int             tc;
    unsigned int    dgd;
    int             channel = 0;
    long            freq[2] = {24576001,19968001};
    char            help[2][80] =
    {
    "４:左  ６:右  ８:拡大  ２:縮小  ＳＰＡＣＥ:サンプリング  ＥＳＣ:コマンド ",
    "Ｌ:読む Ｓ:書く Ｐ:変動率 Ｒ:カウント値 Ｊ:ジャンプ Ｃ:チャンネル Ｑ:終了"
    };
    int     _fmode = O_BINARY;

void printstat(void);
void load(char *fname,char huge *buff,long tfrsize);
void save(char *fname,char huge *buff,long tfrsize);
void hst(char *buff);
void command(char *buff);
void oomoji(char *buff);
void delchar(char *buff,int point,int *len);

void drawmemo(int xmemo,int ymemo);
void dispdata(char huge *buff,int xscl,long datatop,long datanum);
void drawdata(void);
void drawanadata(int zy,int xscl,int *odat,int ndat,int dnum);

void setlinestyle(unsigned int pat);
void line(int x1,int y1,int x2,int y2,int plane);
void setgdc(int *cmd);
void outgdc(int port,int data);
void dispvram(int on_off);
void crtinit(void);
void setpalet(int *pal);

void clearvram(int sy,int cl,int plane);
void vline(int min,int max,int xx,unsigned int pattern,unsigned int dgd);
void sampstart(char huge *buff,int tcount,int dd,unsigned int dgd,int ch);

int main()
{
    char huge       *databuff;
    int             key;
    int             endflg = 0;
    long            disptop,dispmax;
    char            cmdbuff[80];
    int             tcount[2] = {32,26};
    int             xx  = 6;            /*  隣りのデータの間隔      */
    int             xbai[MAX_X_BAI]= {-50,-20,-10,-5,-2,1,2,5,10,20};
    int             xmemo   = 20;       /*  メモリ横方向ドット数    */
    int             ymemo   = 32;       /*  メモリ縦方向どっと数    */
    int             drawflg;

    long            l;

    disptop = 0;
    dispmax = 64;

    crtinit();
    dispvram(1);

    sysclock = (peekb(0,0x0501)&0x80)>>7;       /*  システムクロック    */
    dgd = (peekb(0,0x054d)&0x04)<<4;            /* GDC clock            */

    tc = tcount[sysclock];

    _setcursortype(_NOCURSOR);
    clrscr();
    if ((databuff = farmalloc(MAX_DATA)) == NULL){
        printf("メモリーが確保できません¥n");
        exit(1);
    }

    for (l = 0; l < MAX_DATA; l++)  *(databuff+l)=0;

    setlinestyle(0xffff);
    drawmemo(xmemo,ymemo);

    gotoxy(1,22); cprintf("%s",help[0]);

    while (endflg == 0){
        drawflg = 0;
        printstat();
        key = getch();
        switch(key){
          case ESC :
            command(cmdbuff);
            switch(cmdbuff[2]){
              case 'S' :        /*  セーブ          */
                save(cmdbuff+3,databuff,MAX_DATA);
                break;
              case 'L' :        /*  ロード          */
                load(cmdbuff+3,databuff,MAX_DATA);
                drawflg = 1;
                break;
              case 'P' :        /*  開始変位の設定  */
                hst(cmdbuff+3);
                drawflg = 1;
                break;
              case 'R' :        /*  サンプリング周波数の設定    */
                tc = atoi(cmdbuff+3);
                break;
              case 'J' :        /*  表示位置ジャンプ        */
                disptop = atol(cmdbuff+3)*2;
                drawflg = 2;
                break;
              case 'C' :        /*  ＡＤＣ  チャンネル変更  */
                channel = atoi(cmdbuff+3);
                if ((channel < 0)||(channel > 7))   channel = 0;
                break;
              case 'Q' :        /*  終了            */
                endflg = 1;
                break;
              default :         /*  再表示          */
                drawflg = 2;
                clrscr();
                gotoxy(1,22); cprintf("%s",help[0]);
                printstat();
                break;
            }
            break;
          case ' ' :
            sampstart(databuff,tc,ssd,dgd<<8,channel);
            drawflg = 1;
            break;
          case '6' :                    /*  右半ページスクロール    */
            disptop += dispmax;
            if (disptop+dispmax*2 >= MAX_DATA){
                disptop = MAX_DATA - dispmax*2;
            }
            drawflg = 1;
            break;
          case '4' :                    /*  左半ページスクロール    */
            disptop -= dispmax;
            if (disptop < 0)    disptop = 0;
            drawflg = 1;
            break;
          case '8' :                    /*  拡大    */
            xx++;
            if (xx >= MAX_X_BAI)    xx = MAX_X_BAI-1;
                else                drawflg = 2;
            break;
          case '2' :                    /*  縮小    */
            xx--;
            if (xx < 0)             xx = 0;
                else                drawflg = 2;
            break;
        }

        if (drawflg != 0){
            if (xbai[xx] < 0){
```

```
				dispmax = (GRPH_X_MAX*(-xbai[xx]));
				xmemo = 10;
			}else{
				dispmax = (GRPH_X_MAX/xbai[xx]);
				xmemo = xbai[xx]*10;
			}
			disptop &= 0xfffe;	/* 偶数のみ有効	*/
			if (drawflg > 1)	drawmemo(xmemo,ymemo);
			dispdata(databuff,xbai[xx],disptop,dispmax);
		}
	}

	dispvram(0);				/* グラフィック非表示 */
	clrscr();				/* テキスト消去	*/
	_setcursortype(_NORMALCURSOR);		/* カーソル点滅	*/
	return 0;
}

/*
** ステータス表示
*/
void printstat(void)
{
	gotoxy(1,1);
	cprintf("CH-%1d  周波数 %ld[Hz] ÷%3d =%6ld[Hz]   変動による開始 ±%3u% (±%4u)",channel,freq[sysclock],
tc,freq[sysclock]/tc,ssp,ssd);
}

/*
** ロード
**
*/
void load(char *fname,char huge *buff,long tfrsize)
{
	int	fd,errflg,readsize;
	long	ptr;

	errflg = 0; ptr = 0;
	if ((fd = open(fname,O_RDONLY|O_BINARY)) != -1){
		do {
			if (tfrsize < MAX_INT)	readsize = (int)tfrsize;
				else		readsize = MAX_INT;
			if (read(fd,(char *)(buff+ptr),readsize) == -1){
				errflg = 1;
				break;
			}
			tfrsize -= readsize;
			ptr += readsize;
		} while (tfrsize > 0);
	}else{
		errflg = 1;
	}

	if (errflg != 0){
		gotoxy(1,23); cprintf("disk read error.¥n");
	}
}

/*
** セーブ
**
*/
void save(char *fname,char huge *buff,long tfrsize)
{
	int	fd,errflg,writesize;
	long	ptr;

	errflg = 0; ptr = 0;
	if ((fd = creat(fname,S_IWRITE)) != -1){
		do {
			if (tfrsize < MAX_INT)	writesize = (int)tfrsize;
				else		writesize = MAX_INT;
			if (write(fd,(char *)(buff+ptr),writesize) != writesize){
				errflg = 1;
				break;
			}
			tfrsize -= writesize;
			ptr += writesize;
		} while (tfrsize > 0);
	}else{
		errflg = 1;
	}

	if (errflg != 0){
		gotoxy(1,23); cprintf("disk write error.¥n");
	}
}

/*
** サンプリング開始待ち変動量を%単位で指定する
**
*/
void hst(char *buff)
{
	int	per,max;

	per = atoi(buff);
	if ((per > 0)&&(per < 100)){
		max = 0x0fff;			/* 12ビット最大値	*/
		max = (max/100)*per;
		ssp = per;
		ssd = max;
	}else{
		ssp = 0;
		ssd = 0;
	}
}

/*
** コマンド入力
**
*/
void command(char *buff)
{
	gotoxy(1,22); cprintf("%s",help[1]);
	gotoxy(1,23); clreol();
	_setcursortype(_NORMALCURSOR);
	cprintf(">>");
	buff[0] = 70;
	cgets(buff);
	_setcursortype(_NOCURSOR);
	oomoji(buff+2);
	gotoxy(1,23); clreol();
	gotoxy(1,22); cprintf("%s",help[0]);
}

/*
** 小文字大文字変換
**
*/
void oomoji(char *buff)
{
	int	i,len;

	len = strlen(buff);
	for (i = 0; i < len; i++){
		if (buff[i] == ' ') delchar(buff,i,&len);	/* スペースは削除 */
		if (((buff[i]>0x80)&&(buff[i]<0xA0))||((buff[i]>0xdf)&&(buff[i]<0xfd))){
			i++;
		}else{
			if (buff[i] < ' ')	buff[i] = (char)0;
			if ((buff[i] >= 'a')&&(buff[i] <= 'z')){
				buff[i] = buff[i] - ('a'-'A');
			}
		}
	}
}

/*
** 文字列から1文字削除
**
*/
void delchar(char *buff,int point,int *len)
{
	int	i;

	for (i = point; i < (*len); i++){
		buff[i] = buff[i+1];
	}
	(*len)--;
}

/*
** データを画面に表示する。
**
**
*/
void dispdata(char huge *buff,int xscl,long datatop,long datanum)
{
	long	i;
	int	j,di;
	int	odat,dat;
	int	min,max;
```

```
    clearvram(GRPH_Y_OFF, DATA_RANGE+1, 1);

    if (xscl < 0)       di = (-xscl);
        else            di = 1;

    for (i = 0; i <= datanum; i += di) {
        if (xscl > 0) {                 /*  通常折れ線グラフモード  */
            dat = ((buff[datatop+i*2+1]<<4)|(buff[datatop+i*2]>>4))&0xff;
            drawanadata(GRPH_Y_OFF, xscl, &odat, dat, (int)i);
        }else{                          /*  最大値最小値モード      */
            min = DATA_RANGE; max = 0;
            for (j = 0; j < di; j++) {
                dat = ((buff[datatop+(i+j)*2+1]<<4)|(buff[datatop+(i+j)*2]>>4))&0xff;
                if (dat < min)      min = dat;
                if (dat > max)      max = dat;
            }
            vline(min, max, (int)(i/di), 0xffff, dgd<<8);
        }
    }
    gotoxy(1, 20);   cprintf("%-7lu", datatop/2);
    gotoxy(73, 20);  cprintf("%7lu", datatop/2+datanum);
}

/*
**  メモリを描く
**
*/
void drawmemo(int xmemo, int ymemo)
{
    int         x, y, dx, m10, tenflg, count;
    unsigned    lpat[2] = {0xffff, 0xcccc};

    clearvram(GRPH_Y_OFF, DATA_RANGE+5, 0);

    if (xmemo > 20) {
        dx = xmemo/2;
        tenflg = 2;
    }else{
        dx = xmemo;
        tenflg = 0;
    }

/*  X方向にメモリを描く     */
    count = 0;
    for (x = 0; x <= GRPH_X_MAX; x+=dx) {
        if (tenflg != 0) {                  /*  間隔が広い場合点線を入れる  */
            setlinestyle(lpat[count%2]);
        }
        if ((count%10) == 0)    m10 = 4;    /*  １０本おきに長く    */
            else                m10 = 0;
        line(x, GRPH_Y_OFF, x, GRPH_Y_OFF+DATA_RANGE+m10, 1);
        count++;
    }

/*  Y方向にメモリを描く     */
    setlinestyle(lpat[0]);
    for (y = GRPH_Y_OFF; y <= GRPH_Y_OFF+DATA_RANGE+1; y+=ymemo) {
        line(0, y, GRPH_X_MAX, y, 1);
    }
}

/*
**  データを描く（アナログ用）
**
**  IN
**  zx:     Ｏ点Ｘ座標（ドット）
**  zy:     Ｏ点Ｙ座標（ドット）
**  xscl:   Ｘ方向隣りの点とのドット数
**  yscl:   Ｙ方向１あたりのドット数×１０
**
*/
void drawanadata(int zy, int xscl, int *odat, int ndat, int dnum)
{
    int xx, yy;

    ndat = DATA_RANGE-(ndat);               /*  データを画面表示用に逆転        */
    if (dnum > 0) {
        xx = dnum*xscl;                     /*  Ｘ方向１つ前のデータのドット計算*/
        yy = zy;                            /*  Ｙ方向基準ドットの計算          */
        line(xx-xscl, yy+(*odat), xx, yy+ndat, 2);
    }
    *odat = ndat;
}

/*
**  ＧＤＣのラインスタイルの設定
**
*/
void setlinestyle(unsigned int pat)
{
    int     cmd[16];

    cmd[0] = WRITE;
    cmd[1] = -1;
    setgdc(cmd);

    cmd[0] = TEXTW;
    cmd[1] = pat&0xff;
    cmd[2] = pat>>8;
    cmd[3] = -1;
    setgdc(cmd);
}

/*
**  ＧＤＣで直線を引く
**
*/
void line(int x1, int y1, int x2, int y2, int plane)
{
    int             dx, dy, dmy, cmd[16];
    unsigned int    dir, dc, d, d2, d1, ead;

    if ((x1 > x2)||((x1 == x2)&&(y1 > y2))) {
        dmy = x1; x1 = x2; x2 = dmy;
        dmy = y1; y1 = y2; y2 = dmy;
    }
    dx = x2-x1; dy = y2-y1;

    if (dy > 0) {
        if (dx < dy) {
            dir = 0;
            dmy = dx; dx = dy; dy = dmy;
        }else{
            dir = 1;
        }
    }else{
        dy = -dy;
        if (dx > dy) {
            dir = 2;
        }else{
            dir = 3;
            dmy = dx; dx = dy; dy = dmy;
        }
    }
    dc = dx; d1 = 2*dy; d = d1-dx; d2 = d-dx;

    cmd[0] = VECTW;
    cmd[1] = 0x08|dir;
    cmd[2] = dc&0xff;
    cmd[3] = (dc>>8)|dgd;
    cmd[4] = d&0xff;
    cmd[5] = d>>8;
    cmd[6] = d2&0xff;
    cmd[7] = d2>>8;
    cmd[8] = d1&0xff;
    cmd[9] = d1>>8;
    cmd[10]= 0;
    cmd[11]= 0;
    cmd[12]= -1;

    while ((inportb(GDC_STAT)&0x08) != 0) {  /*  描画が終了するまで待つ  */
        outportb(0x5f, 0);
        outportb(0x5f, 0);
    }

    setgdc(cmd);

    ead = plane*0x4000+y1*40+x1/16;
    cmd[0] = CSRW;
    cmd[1] = ead&0xff;
    cmd[2] = ead>>8;
    cmd[3] = (x1%16)<<4;
    cmd[4] = -1;
    setgdc(cmd);

    cmd[0] = VECTE;
    cmd[1] = -1;
    setgdc(cmd);

}

/*
**  Ｇ－ＧＤＣに対してコマンド＋パラメータ列を送る。
**
*/
void setgdc(int *cmd)
{
    int     i = 1;

    outgdc(GDC_CMD, cmd[0]);
    while (cmd[i] != -1) {
        outgdc(GDC_PARA, cmd[i]);
        i++;
    }
}

/*
**  Ｇ－ＧＤＣに対して１バイトのデータを出力
**
*/
void outgdc(int portadr, int dat)
{
    while ((inportb(GDC_STAT)&0x02) != 0) {         /*  FIFOが空くまで待つ  */
        outportb(0x5f, 0);
        outportb(0x5f, 0);
    }
    outportb(portadr, dat);
    outportb(0x5f, 0);
}
```

```
/*
** パレットを設定する。
*/
void setpalet(int *pal)
{
    int     i;
    int     padr[4] = {0xae, 0xaa, 0xac, 0xa8};

    for (i = 0; i < 4; i++){
        outportb(padr[i%4], (pal[i]<<4)|pal[i+4]);
    }
}

/*
** ＶＲＡＭ表示ＯＮ／ＯＦＦ
*/
void dispvram(int on_off)
{
    union REGS  regs;

    regs.h.ah = 0x40 + (1-on_off);

    int86(BIOS_CRT, &regs, &regs);
}

/*
** ８色 ４００ライン
** ＶＲＡＭ消去 パレット設定
** 描画ＶＲＡＭ表 表示ＶＲＡＭ表
*/
void crtinit(void)
{
    int         i;
    union REGS  regs;
    int         pal[] = {0, 1, 6, 6, 4, 5, 6, 7};

    regs.h.ah = 0x42;
    regs.h.ch = 0xc0;

    int86(BIOS_CRT, &regs, &regs);      /* ４００ライン，カラー， */
    int86(BIOS_CRT, &regs, &regs);

    outportb(0x6a, 0);                  /* ８色モード             */
    outportb(0x5f, 0);
    outportb(0xa4, 0);                  /* 表示画面  ＶＲＡＭ表   */
    outportb(0x5f, 0);
    outportb(0xa6, 0);                  /* 描画画面  ＶＲＡＭ表   */

    for (i = 0; i < 3; i++)      clearvram(0, 400, i);

    setpalet(pal);
}
```

〈リスト 5-2〉 ADC サンプリング・プログラム（アセンブラ部）

```
;
;  ＡＤＣサンプリングプログラムアセンブラ部
;
;  １８６命令使用為  Ｖ３０以降有効
;
        PUBLIC  _clearvram              ; ＶＲＡＭ消去
        PUBLIC  _vline                  ; 最大値－最小値  直線描画
        PUBLIC  _sampstart              ; サンプリング開始

ADC_DATA        equ     0dd0h
ADC_CH          equ     0dd1h
ADC_RESET       equ     0dd2h

GRPH_Y_OFF      equ     40
GDC_RED         equ     8000h

GDC_STAT        equ     0a0h
GDC_CMD         equ     0a2h
GDC_PARA        equ     0a0h

GDC_VECTW       equ     4ch
GDC_VECTE       equ     6ch
GDC_CSRW        equ     49h
GDC_TEXTW       equ     78h

        LOCALS
        .186
        .MODEL  COMPACT
        .DATA
        .DATA?
maskreg         db      ?       ; ８２５９マスタ  ＩＭＲ
timerseg        dw      ?       ; インターバルタイマ旧ベクタ
timeroff        dw      ?
min             db      ?       ; 最小値
max             db      ?       ; 最大値
xx              dw      ?       ; Ｘ座標（０～６３９）
mul40           db      ?
div16           db      ?

        .CODE

;=====================================
;
;  ＶＲＡＭ消去
;
; void clearvram(int sy, int cl, int plane)
;
; sy    : 消去開始Ｙ座標
; cl    : 消去行数
; plane : ＶＲＡＭプレーン
;
_clearvram PROC
        push    bp
        mov     bp, sp
        push    es
        push    di

        call    checkdraw       ; 描画終了まで待つ

        mov     bx, [bp+4]      ; 消去開始行
        mov     ax, 80
        mul     bx
        mov     di, ax

        mov     bx, [bp+6]      ; 消去行数
        mov     ax, 40
        mul     bx
        mov     cx, ax

        mov     bx, [bp+8]      ; 消去ＶＲＡＭプレーン
        mov     ax, 800h
        mul     bx
        add     ax, 0a800h
        mov     es, ax

        cld
        mov     ax, 0
        rep     stosw

        pop     di
        pop     es
        pop     bp
        ret
_clearvram ENDP
;=====================================
;
;  最大値－最小値に垂直方向に線を引く
;
; void vline(int min, int max, int xx, unsigned linestyle, unsigned int dgd)
;
; BX CX DX DI ES はサンプリング部が使用している為変更していけない。
;
_vline  PROC
        push    bp
        mov     bp, sp
        push    si

        call    checkdraw       ; 描画終了まで待つ

        mov     al, GDC_TEXTW   ; ラインスタイルの設定
        call    outcmdb
        mov     ax, [bp+10]
        call    outparaw

        mov     al, GDC_VECTW
        call    outcmdb
        mov     al, 08h         ; 直線，描画方向０
        call    outparab        ; set SL, R, C, T, L, DIR
        mov     ax, [bp+6]
        sub     ax, [bp+4]      ; ax = max-min
        or      ax, [bp+12]     ; ax = ax | dgd
        call    outparaw        ; set DC
        neg     ax              ; ax = -(max-min)
        call    outparaw        ; set D
        rol     ax, 1           ; ax = ax*2
        call    outparaw        ; set D2
        mov     ax, 0
        call    outparaw        ; set D1

        mov     al, GDC_CSRW    ; 描画アドレスの設定
        call    outcmdb
        mov     ax, 255
        sub     ax, [bp+6]      ; ax = 255-max
        mov     [mul40], 40
        mul     [mul40]         ; ax = max * 40
        mov     si, ax          ; ax を一時退避
        mov     ax, [bp+8]      ; ax = xx
        mov     [div16], 16
        div     [div16]         ; ax = xx/40
        push    ax
        mov     ah, 0
        add     ax, si
        add     ax, GDC_RED+GRPH_Y_OFF*40
        call    outparaw        ; set EAD
        pop     ax
        mov     al, ah          ; xx/40 の余り
        shl     al, 4
        call    outparab        ; set dAD
```

```
        mov     al,GDC_VECTE    ; 描画開始
        call    outcmdb
        pop     si
        pop     bp
        ret
_vline  ENDP

;
;  G－GDC描画終了まで待つ
;
checkdraw:
        push    ax
@@notready:
        out     5fh,al
        out     5fh,al
        in      al,GDC_STAT
        test    al,08h
        jnz     @@notready
        pop     ax
        ret

;
;  G－GDCのＦＩＦＯが空くまで待つ
;
;
checkfifo:
        push    ax
@@notready:
        out     5fh,al

        out     5fh,al
        in      al,GDC_STAT
        test    al,02h
        jnz     @@notready
        pop     ax
        ret


;
;  G－GDCにコマンドを送る
;
; IN
;       al      : GDCに送るコマンド
;
outcmdb:
        call    checkfifo
        out     5fh,al
        out     5fh,al
        out     GDC_CMD,al
        ret

;
;  G－GDCに１バイトパラメータを送る
;
; IN
;       al      : GDCに送るパラメータ
;
outparab:
        call    checkfifo
        out     5fh,al
        out     5fh,al
        out     GDC_PARA,al
        ret

;
;  G－GDCに２バイトのパラメータを送る
;
; IN
;       ax      : GDCに送るパラメータ
;
outparaw:
        push    ax
        call    outparab
        mov     al,ah
        call    outparab
        pop     ax
        ret

;=====================================
;
;  サンプリング開始
;
; void sampstart(char huge *buff,int tc,int dd,unsigned int dgd,int ch)
;
; buff  :サンプリングデータ書き込みアドレス
; tc    :サンプリング間隔（インターバルタイマカウント値）
; dd    : = 0 即サンプリング開始
;       :!= 0 初期値から ±dd 変動したらサンプリング開始
; dgd   :GDC clock 5MHz -> 01000000b , 2.5MHz -> 0
; ch    :ADC のチャンネル(0-7)
;
_sampstart PROC
        push    bp
        mov     bp,sp
        push    ds
        push    es
        push    di
        push    si
        call    adcreset                ; ＡＤＣ初期設定

        les     di,dword ptr [bp+4]     ; buff

        mov     cx,[bp+8]               ; tc
        call    settimer                ; タイマー登録

        mov     ax,[bp+10]              ; dd
        call    setstart
        cld
        mov     dx,ADC_DATA             ; adc I/O アドレス設定
        mov     al,0feh                 ; タイマー割り込み許可
        out     02h,al

        call    dispdata                ; インジケータ表示

        call    endtimer                ; ベクターを戻す
        pop     si
        pop     di
        pop     es
        pop     ds
        pop     bp
        ret
_sampstart      ENDP


;
;  ＡＤＣリセット，チャンネル設定
;
; IN
;  [bp+14]      :ADC channel
;
; ax,dx 破壊
;
adcreset:
        mov     dx,ADC_RESET            ; ADC reset
        out     dx,al
        mov     dx,ADC_CH               ; ADC channel set
        mov     ax,[bp+14]              ; ch
        shl     al,5
        out     dx,al
        mov     dx,ADC_DATA             ; ADC dumy read
        in      ax,dx
        call    wait13
        ret

;
;  １３μ秒ウエイト
;
; 破壊 ax
;
wait13:
        mov     ah,26
@@loop:
        out     5fh,al          ; ０．５μ秒
        dec     ah
        jnz     @@loop
        ret


;
;  サンプリング開始待ちの変動範囲の指定
;
; IN
;  ax   : 変動値
;  es di: 書き込みバッファー
; OUT
;  bx   : 最大値
;  cx   : 最小値
;  bx = cx の場合すぐにサンプリング開始
;
; 破壊 ax
;
setstart:
        mov     es:[di],0ffffh  ; 未読み込みマーク

        mov     bx,ax
        mov     cx,ax
        cmp     ax,0
        jz      @@exit

        mov     dx,ADC_DATA     ; ADC 初期値
        in      ax,dx
        and     ax,0fffh
        add     bx,ax
        sub     ax,cx
        jns     @@puls
        mov     ax,0
@@puls:
        mov     cx,ax
@@exit:
        ret


;
;  インジケータ表示
;
; IN
;  es di:書き込みバッファ
;
```

```
; 破壊 ax
;
dispdata:
        cmp     es:[di],0ffffh          ; サンプリングが開始
        jz      dispdata                ; するまで待つ

        mov     [xx],0                  ; x方向座標クリアー
@@minmaxstart:
        mov     [min],0ffh              ; 最小値データクリア
        mov     [max],0                 ; 最大値データクリア
        call    clearline               ; ライン消去
        call    setline                 ; ライン描画

        mov     ax,[xx]                 ; xx をインクリメント
        inc     ax
        cmp     ax,640                  ; 639 越えたら 0 に
        jb      @@setxx
        mov     ax,0
@@setxx:
        mov     [xx],ax

        or      di,di                   ; サンプリングが終了するまで待つ
        jnz     @@minmaxstart

        ret

;
;  描画中に最大値と最小値をチェックする
;
setminmax:
        out     5fh,al
        out     5fh,al
        in      al,GDC_STAT             ; ＧＤＣステータス

        push    ax
        cmp     bx,cx                   ; 開始待ち終了？
        jnz     @@notstart
        mov     ax,es:[di-2]            ; データを読み出す。(ｻﾝﾌﾟﾘﾝｸﾞ中)
        jmp     @@conv256
@@notstart:
        mov     ax,es:[di]              ; データを読み出す。(開始待ち)
@@conv256:
        shr     al,4                    ; データを１２ビットから
        shl     ah,4                    ; ８ビットに変換する。
        or      al,ah
        cmp     al,[min]                ; 最小値かチェック
        jae     @@notmin
        mov     [min],al
@@notmin:
        cmp     al,[max]                ; 最大値かチェック
        jbe     @@notmax
        mov     [max],al
@@notmax:
        pop     ax

        test    al,02h                  ; 描画中か？
        jnz     setminmax
        ret

;
;  インジケータ用１ライン消去
;
clearline:
        call    setminmax
        push    [bp+12]                 ; dgd
        push    0                       ; ライン消去
        push    [xx]                    ; xx
        push    00ffh                   ; max
        push    0                       ; min

        call    _vline
        add     sp,10                   ; スタックを戻す
        ret

;
;  インジケータ用最小値－最大値描画
;
setline:
        call    setminmax
        push    [bp+12]                 ; dgd
        push    0ffffh                  ; ライン描画
        push    [xx]                    ; xx
        mov     al,[max]                ; max
        mov     ah,0
        push    ax
        mov     al,[min]                ; min
        push    ax

        call    _vline
        add     sp,10                   ; スタックを戻す
        ret

;
;  インターバルタイマーをセットする
;
; IN
;  cx   :timer count
;
; 破壊 ax,bx,dx
;
settimer:
        push    ds
        push    es
        in      al,02h                  ; 割り込みマスクレジスタ保存
        mov     [maskreg],al
        mov     al,0ffh                 ; 全割り込み禁止
        out     02h,al

        mov     ax,3508h                ; インターバルタイマの
        int     21h                     ; 旧ベクタを保存
        mov     ax,es
        mov     [timerseg],ax
        mov     [timeroff],bx

        mov     ax,cs                   ; インターバルタイマに
        mov     ds,ax                   ; 新ベクタを登録
        mov     dx,offset sampling
        mov     ax,2508h
        int     21h

        mov     al,36h                  ; タイマー＃０モード３
        out     77h,al
        mov     al,cl                   ; カウンター下位設定
        out     71h,al
        mov     al,ch                   ; カウンター上位設定
        out     71h,al

        pop     es
        pop     ds
        ret


;
; インターバルタイマーを終了する。
;
; ax,dx 破壊
;
endtimer:
        push    ds
        mov     al,[maskreg]            ; 割り込みマスクを戻す
        or      al,00000001b
        out     02h,al
        mov     dx,[timeroff]
        mov     ax,[timerseg]   ; ベクタを戻す
        mov     ds,ax
        mov     ax,2508h
        int     21h
        pop     ds
        ret


;
; サンプリング部
;
; bx    : サンプリング待ち最大値
; cx    : サンプリング待ち最小値
; dx    : ADCデータＩ／Ｏアドレス
; es:di : 書き込みアドレス
;
; 高速化の為 上記レジスタは保存していない。
;
sampling:
        push    ax
        in      ax,dx                   ; サンプリングデータを読む

        cmp     bx,cx                   ; サンプリング開始待ち？
        jnz     @@check
        stosw

        or      di,di                   ; di = 0 でサンプリング終了
        jz      @@finish

@@eoi:
        mov     al,20h                  ; 割り込み終了
        out     00h,al
        pop     ax
        iret

@@finish:
        mov     al,01111001b            ; タイマー割り込み禁止
        out     02h,al
        jmp     @@eoi

@@check:
        and     ax,0fffh                ; １２ビットデータ
        cmp     ax,bx                   ; 最大値より大きいか？
        jg      @@start
        cmp     ax,cx                   ; 最小値より小さいか？
        jl      @@start
        mov     es:[di],ax
        jmp     @@eoi
@@start:
        add     di,2
        stosw
        mov     cx,bx                   ; サンプリング待ち終了
        jmp     @@eoi

        end
```

●本書掲載記事の利用についてのご注意 ── 本書掲載記事には著作権があり，また工業所有権が確立されている場合があります．したがって，個人で利用される場合以外は所有者の承諾が必要です．
　また，掲載された回路，技術，プログラムを利用して生じたトラブル等については，小社ならびに著作権者は責任を負いかねますのでご了承ください．

●本書に関するご質問について ── 文章，数式等の記述上で不明な点についてのご質問は，必ず往復はがきか返信用封筒を同封した封書にてお願いいたします．ご質問は著者に回送し直接回答していただきますので，多少時間がかかります．また，本書の範囲を超えるご質問には応じられませんので，ご了承ください．

R ＜日本複写権センター委託出版物＞
本書の全部または一部を無断で複写複製(コピー)することは，著作権法上での例外を除き，禁じられています．本書からの複写を希望される場合は，日本複写権センター(☎03-3401-2382)にご連絡ください．

トランジスタ技術 SPECIAL No.45

©CQ出版㈱ 1994
©吉田 功 1996

1994年5月1日 初版発行
1996年4月1日 第2版発行

編集人
発行人 蒲生良治
発行所 CQ出版株式会社 〒170 東京都豊島区巣鴨1-14-2
電話 03-5395-2123(出版部)，03-5395-2141(販売部)
振替 00100-7-10665

(定価は表四に表示してあります)

印刷・製本 三晃印刷株式会社

乱丁，落丁本はお取り替えいたします

Printed in Japan

ハードウェア・デザイン・シリーズ　好評発売中

電池活用ハンドブック

トランジスタ技術編集部 編
B5判 208頁 定価1,800円



電池の知識－電池活用デバイス－電池活用回路－充電回路－電池活用資料集

本書は，各種電池の基礎知識を身につけるための解説記事，電池を活用するためのデバイスの使い方と回路設計技術，充電器・測定器の製作記事，小型電池の資料集などから構成されています．

実用電源回路設計ハンドブック

戸川治朗 著
B5判 240頁 定価1,960円

整流回路からスイッチング・レギュレータまで

本書は，電源回路の設計法を，定数の求め方を中心に紹介した初めての本です．シリーズ・レギュレータから，スイッチング・レギュレータの設計法，放熱やノイズ対策への話題も豊富です．

電子回路部品活用ハンドブック

トランジスタ技術編集部 編
B5判 288頁 定価1,850円

受動部品/機構部品を100%活用するために

基本的な受動部品である抵抗／コンデンサ／インダクタをはじめ，コネクタ，リレーなどの機構部品，電池，パワー・サーミスタなどの電源回路部品，EMIフィルタ，バスバーなどのノイズ対策部品について解説．

メカトロ・センサ活用ハンドブック

トランジスタ技術編集部 編
B5判 224頁 定価1,850円

FA現場でのセンサ活用から応用回路設計まで

各センサを使いこなせるように，動作原理・使用上の注意のほか基本回路と応用回路についても詳解しました．

新つくるシリーズ　●B5判●160頁●各 定価 1,500円●

新つくるシリーズ

エレクトロニクスのわかりやすい入門書が欲しいという声をよく耳にします．しかし，万人に対してわかりやすいというテーマを実現することは簡単ではありません．

「わかりやすい」ということを実現することはたいへんなのですが，エレクトロニクスについては「こうやって学べばよいのではないか」という答えがあります．それは，「自分の手で作ってみる」ということです．天才は閃きで物事を解明していくことができるかもしれませんが，凡人にとっては人真似から入るのも合理的です．

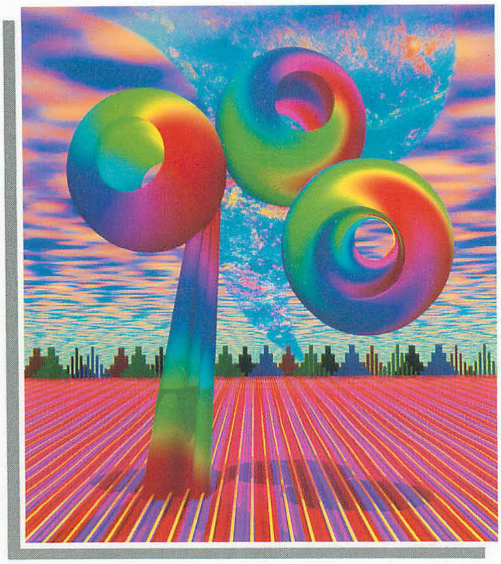
ということで用意したのが，この３冊の「新つくるシリーズ」です．いずれも『トランジスタ技術』誌，および『トラ技ORIGINAL』誌で掲載され，好評を博した記事のなかから，つくりたくなる記事，つくることを擬似体験できる記事をジャンルごとに再構成しました．真似をして体験することが最高の学習になると思いますが，読んでいるだけでも利用できそうなアイデアをふんだんにカバーしています．

No.1　〈好評発売中〉
つくるツール＆測定器
おもな内容●ディジタル電圧計／ファンクション・ジェネレータ／カーブ・トレーサ／LCメータ／etc.

No.2　〈好評発売中〉
つくるオーディオ＆ビデオ
おもな内容●オーディオ・アンプ／サウンド・プロセッサ／ビデオ・セレクタ／ビデオ・エフェクタ／etc.

No.3　〈好評発売中〉
つくるオリジナル・グッズ
おもな内容●電子ゲーム／キッチン・タイマ／電子温度計／電磁波時計／ニカド電池充電器／紫外線メータ／etc.

CQ出版社 〒170 東京都豊島区巣鴨1-14-2 販売部☎(03)5395-2141 振替 00100-7-10665



CQ出版社　定価1,600円(本体1,553円)　ISBN4-7898-3237-6 C3055 P1600E